

UNIVERSITY OF
ILLINOIS LIBRARY
AT URBANA CHAMPAIGN
ENGINEERING

NOTICE: Return or renew all Library Materials! The Minimum Fee for each Lost Book is \$50.00.


JUN 27 1988

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.
To renew call Telephone Center, 353-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

L161—O-1096



Digitized by the Internet Archive
in 2012 with funding from
University of Illinois Urbana-Champaign

<http://archive.org/details/stateoftheartrep150belf>

84
.3c
50

Engin

CONFERENCE ROOM

ENGINEERING LIBRARY
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS

Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

CAC Document Number 150

A State-of-the-Art Report
on Network Data Management
and Related Technology

April 1, 1975

The Library of the
MAY 5 1976
University of Illinois
Urbana-Champaign

A State-of-the-Art Report on
Network Data Management
and Related Technology

by

Geneva G. Belford
Steve R. Bunch
John D. Day

with

Peter A. Alsberg Enrique Grapa
Deborah S. Brown David C. Healy
John R. Mullen

Prepared for the
Joint Technical Support Activity
of the
Defense Communications Agency
Washington, D.C.

under contract
DCA100-75-C-0021

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

April 1, 1975

Approved for release:



Peter A. Alsberg, Principal Investigator

Table of Contents

	Page
Introduction	1
Purpose	1
Notes on the References	1
Data Management	3
Overview	3
Tabular Data Structures	5
Introduction	5
Indexing	5
Clustering and Ordering of the Records	6
Hashing (Key-to-address Transformation)	6
Recent Research	7
Improved Indexing Schemes	7
Optimal Selection of Indexes	7
Combination of Hashing and Inversion	8
Overviews and Analyses of Techniques	8
Relational Forms	8
Automatic, Adaptive Structuring	10
Summary and Assessment	11
Complex Data Structures	13
Introduction	13
Recent Research	15
Optimal Trees	15
Organization of Structure Types	15
Modeling and Evaluation	17
Automatic Structuring	18
Summary and Assessment	19
Hashing	21
Introduction	21
Recent Research	22
Summary and Assessment	24
Clustering and Partitioning	25
Introduction	25
Recent Research	25
Blocking According to Access Frequency	25
Clustering for Document Retrieval	26
Other Clustering Methods Based on Similarity of Keys	27
Clustering Based on Simultaneous Retrieval	27
Summary and Assessment	28
Compression	31
Introduction	31
Logical Compression	31
Null/Blank Suppression	31
Statistical Encoding	32

	Page
Recent Research	32
Evaluation and Experimentation	32
Suboptimal Codes	32
Automatic Analysis and Compression	33
Summary and Assessment	33
Data Languages	35
Introduction	35
Some Recent Work	36
Data Definition Languages	36
Data Manipulation Languages	36
Query Languages: Algebraic	37
Query Languages: Structured English	37
Query Languages: Natural	38
Summary and Assessment	39
Integrity	41
Introduction	41
Recent Work	41
Consistency Checks	41
Maintenance of Backups	42
Concurrent Use	42
Summary and Assessment	43
File Allocation in a Network	45
Introduction	45
Recent and Current Research	45
Optimal Allocation by Zero-One Programming	45
Optimal and Suboptimal Allocation by Search Procedures	46
Some Current Work	47
Summary and Assessment	47
References	49
Network and Systems Environment	57
Overview	57
Communications and Networks	59
Introduction	59
Communication Media	59
Introduction	59
Common Carrier Communication Utilities	60
Telephone Network	61
Datran	61
SPCC - Southern Pacific Communications Company	61
WTCI - Western Telecommunications Incorporated	61
Value-Added Carriers	62
PCI - Packet Communications Incorporated	62
Telenet Communications Corporation	62
Communication Network Design Issues	63
Delay	63
Bandwidth	63

	Page
Reliability	64
Cost	64
Packet Switching	64
Forwarding Schemes	65
Packet Size	65
Store-and-Forward Switching Node Design	66
Topology of Communication Networks	66
Routing	70
Flow Control	72
Network Analysis and Modeling	73
Queuing Theory	73
Network Models: Mathematical Analysis by Queuing Theory and Other Techniques	74
Network Models: Simulations	76
Multiple-Access Broadcast, Radio, and Satellite Communications	77
Introduction	77
Applications of Broadcast Technology	77
Current Technology	78
Broadcasting Media	78
Wire or Cable	78
Radio and Satellites	79
Multiple-Access Techniques and Analysis	80
Ground Radio Techniques	80
Satellite Techniques	81
Summary and Assessment	81
References	83
Resource Allocation and Control	91
Introduction	91
Multiple Copies of Data Bases in a Distributed Environment	91
Multiple-user Problems	91
Multiple Data Base Problems	91
Unreliability and the Reconciliation of Copies	92
The Overall Problem	92
Naming and Locating Resources	93
Name Spaces	93
Naming in a Network	95
Mutual Exclusion	95
Deadlock	96
Deadlock Prevention	97
Deadlock Detection and Recovery	97
Accessing Remote Resources - Protocols	98
Introduction	98
Establishing Communications	98
Telecommunications Network Protocols (Telnet)	100
File Oriented Protocols	101
Load Sharing Techniques	102
Remote Job Entry Protocols	103
Protocol Formalisms	104
Distributed Operating Systems	105

	Page
Summary and Assessment	106
References	109
Measurement and Analysis	115
Introduction	115
Performance Evaluation	116
Analytical Approach	116
Black Box Approach	117
Load Determination and Prediction	117
Summary and Assessment	118
References	120
Network Access Systems and Front-Ends	123
Introduction	123
Major Issues and Research	124
Summary and Assessment	126
References	128
Security	129
Introduction	129
User Authentication	131
Password Generation	131
Password Input and Checking	131
Password Give Away	132
Continuous Authentication	132
Physiological Passwords	132
Data Security	133
Capabilities	133
Domains	133
Implementations	134
Capability Lists	134
Access Control Lists	134
Password per Object	134
Rings	135
Confinement/Leakage	135
Information Leakage	135
Trojan Horse	135
The Census Problem	136
Encryption Techniques	136
Certification	138
The Cost to Break the Security of a System	138
Secure System Technology	138
Provably Secure Systems	140
Security Kernels	140
Summary and Assessment	141
References	142
Network Application Support	145
Overview	145

	Page
User Support	147
Introduction	147
Documentation and Consulting	147
Network-Oriented Documentation Support	148
User-user Facilities	150
Mail	150
Teleconferencing	151
A Network Information Center	152
User Management Facilities	153
Summary and Assessment	153
Installation and Project Support	155
Network Support	156
References	157

Introduction

Purpose

The Center for Advanced Computation of the University of Illinois at Urbana-Champaign is preparing a three year research plan to develop network data management and resource sharing technology for application in the World Wide Military Command and Control System (WWMCCS) intercomputer network. This work is supported by the Joint Technical Support Activity of the Defense Communications Agency.

As part of the preparation of the research plan, the state-of-the-art in network data management and resource sharing was surveyed. An extensive literature search was undertaken and eight active research sites were visited. The state-of-the-art as represented in the literature and at the research sites is summarized in this report.

The report is divided into three main sections: Data Management, Network and Systems Environment, and Network Application Support. Each main section is preceded by an overview and followed by discussions of individual areas in that section. The discussions are tutorial when possible (e.g., the discussion of network front ends). In some cases, the art is so advanced that a tutorial would be a massive undertaking (e.g., data structures). In those cases, the important results are described and the reader is directed to the appropriate technical literature and to the better tutorials (if they exist). It is hoped that the reader of an area discussion and the corresponding section overview will understand the relation of that area to other areas in the section, the state-of-the-art in that area, and the research needs in that area. The individual discussions are intended to be self contained so that the reader need only peruse the overviews and areas of direct interest to him. Occasionally, the work in one area will depend heavily on the results of another area. In those cases, the results are only explained in the primary area. The reader of a dependent area is directed to the appropriate section if he is interested in a more complete understanding.

Notes on the References

Every attempt has been made to make the references complete and correct. For the sake of brevity, certain identifying information has been left out of the reference listings and is included here instead. Standard abbreviations are used; for example, ACM is the Association for Computing Machinery and the frequently referenced journal CACM is the Communications of the ACM. The various ACM proceedings, workshops, etc. are generally published by the ACM, as are publications of the ACM Special Interest Groups (SIGME, SIGFIDET, etc.) and the ACM CODASYL Groups.

The AFIPS Conference Proceedings have in recent years had several publishers: Spartan Books, Washington, D.C. (Vols. 22-29, or 1962-1966), Thompson Books, Washington, D.C. (Vols. 30-33 or 1967-1968) and AFIPS Press, Montvale, N.J. (Vols. 34 or 1969-present). Sometimes the AFIPS Conference Proceedings are identified by type and year instead of by volume number; i.e. SJCC (Spring Joint Computer Conference), FJCC (Fall Joint Computer Conference), or NCC (National Computer Conference).

Much of the technology is described in informal notes. Documents identified as RFC #, NIC #, A.S.S. #, or Packet Radio (PR) # are all informal communications among ARPA network researchers. The official repository for these documents is the Augmentation Research Center, Stanford Research Institute, Menlo Park, Ca. Also referenced are a number of notes from the Internetwork Working Group (INWG). Copies of these may be obtained from Prof. V. Cerf, Digital Systems Lab, Stanford University, Palo Alto, Ca.

Those references followed by the statement "(Indirect Reference)" are references which are included for completeness, but which have not necessarily been read or verified by the authors. These references are given as they appeared in the referencing document, paper, abstract, etc., and may be incomplete or incorrect.

Finally, the reader must take note of the fact that the references are not all in one place but are listed following the appropriate sections. Data Management and Network Application Support have single reference lists. Network and Systems Environment has a separate listing for each of its five areas.

Data Management

Overview

The problems of data management are many and varied. To the uninitiated, storage and retrieval of data may seem to present little difficulty. As data bases grow larger, however, sophisticated techniques must be devised to handle them efficiently. The development of computer networks has led to the prospect of distributed data bases, for which all of the single-site data management problems will become even more complex and additional problems will arise [Chandra, 1973].

Although in this report we are attempting to summarize the general state of the art of data management, we are particularly concerned with recent work on problems that are pertinent to very large data bases in a network environment. We hope that much of this work will be applicable to - or at least point in the direction of - solutions to the important problems of distributed data base management.

The first sections of this report deal primarily with the structures for data storage. The simplest way to store a set of data is as a tabular list, and we have devoted one section to this storage method. The key problem is that of devising efficient methods for finding items in a very large table. We discuss various indexing methods and (in a separate section) hashing, the scheme which does away with a physically stored index and replaces it with an algebraic formula to transform names into addresses.

A simple index is just a name vs. address list. Searching such a list can be very time consuming. One may improve the situation by building indexes for the index, or by arranging the index in a hierarchical or branching structure for more efficient searching. Indeed the data itself may be structured hierarchically - either because it is natural to do so or for search efficiency. Such complex data structures are considered in another section of this report. The reader who wishes to delve deeply into the subject of data organization should note that our emphasis is on recent and current work. For breadth of background a number of standard texts are available, such as [Lefkovitz, 1969], [Flores, 1970], and [Salton, 1968].

No matter how the data is structured, in a very large data base some preliminary grouping of items is in order. That is, items that are frequently retrieved together should be stored together (e.g., on the same tape). This problem, which could become particularly important for a distributed data base (in which parts of the base are to be put at different sites) is discussed under Clustering and Partitioning, below.

As another important aid to handling huge amounts of data efficiently, compression is discussed. The idea here is that compaction of data items (basically through some sort of encoding) can save enormous amounts of storage space. With this savings comes additional savings from shorter search times, less frequent transfers between auxiliary and main memory, etc.

The reader may note one recurrent theme running through the discussions below. This theme is the need for more automation - the development

of good algorithms so that the computer can automatically compress, cluster, structure, and even index the data. There is no question but that this is the direction that future research will (and needs to) go in order to accomplish real breakthroughs in the management of large, distributed data bases.

Following the discussion of structuring and closely related topics, we briefly survey languages for handling data - from the low-level languages for defining physical data structures through the interesting experiments in resolving ambiguities in natural English so that it can be used to input queries. The subject seems currently to be in a state of flux. Controversies exist over the structuring of language levels (which levels should handle which tasks) and how close to "natural" English a query language really needs to be. It seems clear, however, that devising reasonably convenient languages to get the jobs done is not a serious problem.

Next we take up the topic of data integrity. Data may be accidentally destroyed or altered by hardware or software errors. It is important that better methods be developed for reducing such deterioration to a minimum, as well as periodically checking for its occurrence. Topics covered include attempts to develop consistency checks and the problems of concurrent use, particularly in a network environment.

Finally, the last section is devoted to file allocation. Studies of this problem began with the development of multiple memory devices, when it became important to determine which files are best stored on which devices. With the advent of computer networks, the problem became one of determining which files should be stored at which sites. It is the development of methods to attack this latter problem that primarily concerns us here.

Tabular Data Structures

Introduction

The simplest way to store information is in tabular form. A tabular file consists of a list of records, each record containing an identifying number (or name) and a set of attribute values (or key values). Another way of looking at a record is just as an n-tuple of related information items. For example, a personnel file usually consists of a list of records, each record being identified by the person's name (or perhaps social security number). The remainder of each record indicates the person's job, salary, experience, personal data, etc. Actual identification of the items in the record may be either implicit (i.e., some separate file description specifies lengths of fields and the meaning of their contents) or explicit (i.e., the records contain attribute, attribute-value pairs (such as salary = \$10,000), so that no external identification is needed.) The latter approach allows for more flexibility, but the former saves space and is generally used.

It should be noted that such a list of records need not actually be stored in contiguous physical locations. It may be that each record has a pointer to (i.e., contains the address of) the next record in the table. This arrangement is known as a linked-list structure. It is easy to add more records to such a file without having saved in advance a large block of contiguous storage.

The straightforward way to query data in tabular form is to make a record-by-record search down the list until the desired items are found. This process is very time consuming for large files. Much thought and effort have gone into the devising of efficient ways to retrieve data from tabular files. Before discussing recent research trends in the area, we shall briefly review the basic aids to retrieval.

Indexing. Just as one indexes a book for rapid retrieval of information, one can index a file - i.e., generate a separate file which lists record identifiers and locations where the corresponding records may be found. Such a list is also known as a directory. Indexes are also constructed for attributes other than simple record identifiers. Thus, in a personnel file, there may be an index for job types which lists pointers to all persons doing specific kinds of work. Or there may be a listing of pointers to all married employees, or to all employees in certain salary ranges, etc. Indexes of this kind, which provide a mapping from attribute or key values to records possessing those key values are also known as inverted files. One also sees the term "secondary index," referring to an inverted file indexing secondary keys (such as salary or marital status in the example above). The primary key is the unique record identifier (e.g., a name or i.d. number.)

Availability of a large set of secondary indexes aids considerably in answering complex queries. Suppose that one wants to retrieve from a personnel file a listing of all engineers with an M.S. degree. One then

looks up all persons (actually a list of pointers to records) who are engineers in the job index and all persons who have an M.S. in the education index. The intersection of the two lists thus obtained will yield the desired group.

Secondary indexes are expensive, however, not only because of storage costs but because file updates and insertions typically require that changes be made in the indexes as well as the files. Much thought has been given to the study of tradeoffs between retrieval efficiency and update costs (and efficiency). This problem is implicit (if not explicit) in most of the research to be discussed below.

Clustering and Ordering of the Records. The idea here is that rearrangement of the tabular entries may lead to more rapid retrieval. By clustering, we mean the grouping together of records related in some manner. This topic is discussed in detail in a separate section of this report. The notion that ordering records aids in retrieval is an obvious one and the basis of the alphabetical ordering of dictionaries, telephone directories, etc. Ordering is particularly useful if queries involving greater-than or less-than relations are common. For example, if all employees with salaries greater than \$20,000 are requested, it is helpful to have a listing of employees in order of increasing salary.

Ordering is not of much help, however, if one is looking for a particular record and can only search linearly down the list. One usually combines ordering with a certain amount of indexing to expedite retrieval. For example, an alphabetically ordered file may have an associated index which points to the first record beginning with A, B, etc. More complex indexes, in which one looks up the first letter (A), then branches to find the second (AC), etc., are also possible. These will be considered in more detail in the section of this report dealing with complex data structures.

Another method of searching ordered lists is by binary search (binary chopping). One examines the record in the middle of the file to determine whether the desired record is before or after the midpoint. One then examines the midpoint of whichever half of the file has been determined to hold the desired record on the first step. This locates the record in a specific quarter of the file. This process is continued and the desired record is located very quickly. Notice that binary search will not work on a linked-list structure since there is no way to find the central points. Recently a suggestion has been made [Berman and Colijn, 1974] that files be structured into a sequence of linked blocks, each block consisting of ordered records and being stored in contiguous physical locations. In this way the blocks may be searched by the efficient binary technique, but at the same time the large amount of contiguous storage required for binary search of the whole file is not needed.

Hashing (Key-to-address Transformation). A certain amount of storage and search of indexes can be eliminated by hashing, which is the technique of directly computing the address of a record from its identifier or key value. Much work has been done on this technique and it is discussed in detail in a separate section of this report.

Recent Research

Improved Indexing Schemes. The most commonly used approach to making retrieval efficient has been the construction of inverted files or secondary indexes. A secondary index, as described above, consists of a listing of all possible values of some attribute or key with pointers to the records having that attribute value. To respond to a complex query involving several attributes the system must search several secondary indexes and concatenate the resulting lists of pointers. Retrieval might be made more efficient by constructing secondary indexes which list combinations of attribute values and associated record pointers.

Several workers have pursued this idea. We mention only two of the more recent contributions here. Lum [1970] points out that a full multiple-attribute index scheme has the disadvantage that queries only specifying values for a subset of the attributes require retrieval procedures almost as complex as for single-attribute indexing. To clarify the problem, suppose the records have 3 attributes, each of which may take on any of 10 values. The multi-attribute index will then contain 1000 entries, one for each possible triple of values for the 3 attributes. Suppose the index is ordered so that for each block of a hundred entries attribute 1 is fixed while attribute 2 changes every 10 entries, and attribute 3 cycles through its values 1, ..., 10, 1, ..., 10, etc. Now it is relatively easy to retrieve all records for which attribute 1 takes on its 2nd value. The pointers needed for retrieval are contained in one consecutive block. But if we wish to retrieve records for which attribute 3 takes on a particular value, we must search through every tenth entry in the index. Lum's suggestion is that redundant entries be added to the index, the redundancy being in the form of listing the multiple keys with different orderings, to make retrieval by various key subsets equally convenient. The difficulty is, of course, that multiple-attribute indexes are already large and introducing redundancies may make them too large for practical use.

Optimal Selection of Indexes. Pointing out these high storage costs, Stonebraker [1974] discusses the problem of selecting some optimal subset of combined indexes to be actually included in the secondary index. Stonebraker also analyzes the problem of selecting the particular subset of keys to be included in a partially inverted file. His approach is to carefully specify the allowable query set, as well as assumptions on retrieval and update procedures and general index structure. He then proves a number of theorems on the best selection of indexes - best in the sense of minimizing expected interaction time.

A recent report [Yue and Wong, 1974] has appeared which questions the unconstrained minimization of interaction time as a basis for index choice. The approach of Yue and Wong is to put an upper limit on storage cost and minimize query time under this constraint. They consider a multi-level storage hierarchy, with different costs for each level, so that file allocation is considered along with index choice in an overall optimization. The mathematical programming problem obtained is fairly complex; an algorithm is proposed for its solution.

Combination of Hashing and Inversion. Motivated by the same considerations as is multiple-key indexing, work on multiple-attribute hashing has also been carried out [Rothnie and Lozano, 1974]. However, Rothnie and Lozano feel that only certain attributes should be included in the hashing scheme, while other attributes are best handled by simple inverted files. They therefore propose combining the two techniques. To provide a rigorous basis for decisions on which attributes should be handled in which way, they set up a mathematical optimization problem. They assume a paged memory environment, and minimize expected number of page accesses. They claim that even suboptimal solutions obtained by heuristic methods provide index/hashing combinations which "often perform substantially better than the more traditional approach, inversion."

Overviews and Analyses of Techniques. Except for the attempts towards multi-attribute indexing (discussed above), the use of indexing and secondary files has been well developed for many years. Much of the recent work has, therefore, been to put the old ideas into some sort of perspective. Bloom [1969] provided one of the earliest such analyses. He discussed most of the basic indexing and retrieval techniques and evaluated them in terms of costs, retrieval time savings, etc. Most important, he put the benefits of inverted file organization on a solid theoretical basis.

The following year a very influential paper was published [Hsaio and Harary, 1970]. In an attempt to standardize terminology, the notions of record, file, index, key, etc. were formally defined. A generalized file structure was then proposed which encompasses, as special cases, such structures as inverted files (where the directory contains one or more entries for each record) and multilist files (where the directory contains pointers to only the initial records of linked-list files). Hsaio and Harary also discuss in detail two general retrieval algorithms for their structure. The terminology and viewpoint contained in this paper have been adopted by a good many of the researchers working in data management.

Relational Forms. An even more influential attempt to introduce a formal, logical basis to tabular data structures has been the work by Codd (and his many followers) on relational forms. (See, for example, [Codd, 1971], [Codd, 1974], and numerous references therein.) In Codd's terminology, a file becomes a relation, which consists of a set of n-tuples (or just "tuples") each tuple being the usual listing of attribute values which make up a record. In relational jargon, the attributes are the domains of the relation, so that a listing of n-tuples does define a relation in the usual mathematical sense. Codd felt that by introducing formal mathematical underpinnings for the simple tabular structure, he could promote that structure as a general, machine independent, data independent, easily visualized mode of data organization.

Much of the earlier work on relational forms was concerned with how the logical relationships among data items (which are often naturally hierarchical) may be put into simple tabular (or normalized relational) form. Essentially, a normalized relation is characterized by the following five features:

- 1) All rows are distinct (i.e., there is no duplication of records, and each record, being unique, is essentially its own identifier).

2) The ordering of the rows is immaterial. This makes it easy to insert and delete records.

3) All columns have homogeneous elements. This is just the usual tabular data requirement that an "age" field contain an integer age, a "sex" field contain M or F, etc.

4) If the columns are assigned distinct names, their ordering is immaterial. Again, this is an old idea. If the attribute name is attached to its value, the order of the listing doesn't matter.

5) Each column is simple; i.e., it is not a relation. This is a particularly useful feature of the normalized relational form. It has been fairly common for data that starts as a simple table to become complex as natural hierarchies become inserted. For example, a personnel file may contain a listing of schools attended, with under each school a listing of areas studied, and under each area a list of grades. One no longer has a simple table to search but a complicated mix of tabular and hierarchical data. It requires considerable effort and specialized knowledge to find one's way about in such a data structure. Codd's insistence on keeping the tables simple (by adding new tables for each new relation, and including cross references as needed) goes a long way towards achieving the goals of standardized format with standardized retrieval methods. All files (relations) look alike and may be handled in the same way.

Since the data is in relational form, it is natural for a set-theoretic approach to queries and retrieval to be developed. One might ask, for example, for the set of all employee names corresponding to employees whose salaries are between \$10,000 and \$20,000. The salary column (domain) in the personnel file (relation) is then checked against the prescribed range, and whenever the given qualification is satisfied, the content of the name column is returned. One sees that this whole process is completely independent, as is desired, of the fact that we are talking about "employees" and "salaries". Retrieval is just a matter of looking up the appropriate relation table and the appropriate columns in that table.

The problem with relational forms is that they were designed to describe logical relations between data items, without regard to access. The basic retrieval scheme described above is just sequential search, as time-consuming as ever. If it were possible to examine a whole column simultaneously - something like the old punch-card-and-needle hand filing system - the relational approach would be very efficient for retrieval. A good bit of thought has actually been given to the design of such devices, known as associative memories. (See, for example, [Moulder, 1973] and Linde et al., 1973].) An associative memory is basically a parallel processor with the capability of carrying out simple operations on a very large number of items simultaneously and very rapidly. Such a processor could examine a whole column at a time in a relation and immediately retrieve the items matching a query. Or the set operations on relation domains needed to respond to complex queries could be carried out almost instantaneously. The paper by Linde et al. describes an investigation into the potential of associative memories. Simulation studies were carried out, so that the report quantifies the enormous savings (several orders of magnitude) which may be achieved. By attaching a disk with parallel read heads to an array processor, Moulder has actually put together a working associative memory and has implemented an experimental data management system on it. But on the whole, associative memories are still very much in the development stage.

With present technology, therefore, efficient retrieval demands the building of indexes, inverted files, etc., on top of a relational data base, and so the beautiful simplicity is destroyed. Nevertheless, much work is currently being done on implementation of and experimentation with relational data bases. For example, there is a large on-going effort at IBM Research (San Jose). Their approach is to use pure relational forms. They store only simple tables, search n-tuples sequentially, and have developed a rather awkward, set-theoretic query language which requires the user to institute searches in one table after another in order to answer all but the simplest queries [Astrahan and Chamberlin, 1974; Chamberlin and Boyce, 1974]. Other organizations engaging in fairly extensive efforts towards relational data base implementation are the University of California (Berkeley), General Motors, and the University of Toronto [Schuster, 1974]. The work is all very preliminary at this stage.

In an early paper [Codd, 1970], Codd suggested that the relational form might be particularly useful as an aid to maintaining data integrity. One reason for this is that a certain amount of redundancy is almost inevitable in a normalized data base. Some relations may be derived from others by logical or set operations. These interconnections can be specified to the system as constraints, and the system can make periodic consistency checks - i.e., checks to see that the constraints are satisfied. The homogeneity of the elements in each column is also an aid to accuracy checks. Information on the allowable range of each domain (i.e., the allowable values for each attribute) may be given to the system so it can identify gross input errors. There is an on-going effort at IBM Research (San Jose) to develop consistency checks, but the schemes devised so far are limited to this ad hoc imposition of constraints based on a priori knowledge of the data [Traiger, 1975]. Finally, the very simplicity of the relational data base makes it easier for integrity to be maintained. New records or files are always inserted in a standard way. It is much easier for the user to make mistakes during insertion, deletion, or update in an hierarchical or other complex data structure.

Automatic, Adaptive Structuring. As data bases grow in size, it becomes increasingly important that the power of the computer be applied to structuring the data. Not only does automatic structuring take the burden of structuring off the user or data manager, but it also allows dynamic structuring to take place; i.e., periodic restructuring which adapts to the users' needs.

One interesting experiment in automatic data organization has been described in a recent series of three papers [Stocker and Dearnley, 1973; Dearnley, 1974 (a), (b)]. The basic file organization assumed is tabular. The records may be stored sequentially (i.e., in physically consecutive locations) with or without ordering on some key and with or without an additional index. Alternatively the records may be stored randomly by a key-to-address transformation. The self-organizing system has modules for converting the data from one structure to another and also supports a choice of access methods, primarily binary search, serial search, and access through indexes or key-to-address transformation. As queries are entered into the system, search statistics are collected. Periodically these statistics are analyzed and a determination is made as to whether the system would respond more efficiently (assuming present trends continue) if a different organization of the file were available. If justified by a cost analysis,

a new organization (e.g., a sorting or indexing by another key) may simply be generated and stored in addition to already existing file versions. An interesting feature of the system is that, once more than one file/access system has been constructed, each query is examined and an optimal retrieval strategy is determined for it. It should be noted that Dearnley actually has a model system implemented and in working order. Experimentation on the system should provide useful information to guide future research in this area.

A less flexible adaptive system has been proposed by Reardon [1974]. He includes only inverted-file access. The system proceeds by examining each query as it is entered to see whether it can be answered through use of the currently existing inverted file. If a query contains an attribute-value that is not in the index, a sequential search of the file is carried out. The pointers obtained are then entered into the inverted file for future reference. In this way one may start with an empty inverted file and build one dynamically as the system is used. Continual query monitoring is recommended so that entries no longer used may be deleted. Reardon's discussion is hypothetical; he has not implemented such a system.

Summary and Assessment

Even though the basic tools for working with data in tabular form have been well developed for some time, much needs to be done on improving efficiency. The recent interest in determining optimal choices of keys to be indexed [Stonebraker, 1974; Yue and Wong, 1974] holds promise for the future, as does the work [Rothnie and Lozano, 1974] on selecting an optimal combination of hashing and indexing. It is undoubtedly worthwhile for more effort to be put into extensions to and trials of such optimization schemes. It may be, however, that the mathematical minimization problems that these schemes involve are too formidable for use in the context of large data bases.

Automatic restructuring is certainly of importance for the future handling of large amounts of data in an environment where the query patterns are changing. Dearnley's recent implementation of a rather versatile system of this type [Dearnley, 1974 (a), (b)] has shown that dynamic file organization is feasible. Much more work can (and undoubtedly will) be done on this problem in the future.

There is enormous current interest in relational data bases, because many persons feel that relational forms will be useful in solving the problems of data management. Relational forms certainly make insertion of new information easy. They do not, however, obviate the need for indexing or hashing to aid in efficient retrieval and update. (One must find an item in order to update it!) It is perhaps significant that no large data base has yet been put into relational form. Until this is tried, the value of the relational approach is simply not demonstrated.

As noted above, the use of relational forms may aid in maintaining data base integrity. A systematic and thorough study of the possibilities here could well lead to something better than the current ad hoc techniques, but it is clear that identification of bad data cannot be 100 per cent reliable unless the system is given equivalent correct information to check the data against. The ad hoc specification of ranges for the various domains

can, of course, let errors slip through. An automated version of this technique, such as the statistical analysis of columns followed by identification of probable errors, might work equally well or even better.

The simplicity of the relational form makes it attractive as a prospective canonical form for data exchange between different sites on a network. Algorithms for converting complex data formats into normal relational form have been worked out [Codd, 1971]. The conversion process is also said to be reversible. To the best of our knowledge, however, no one has yet successfully implemented a procedure for information transfer which makes use of relational form intermediates. Use of the relational form as a standard throughout a network would make conversion between formats unnecessary and could be a considerable aid to data exchange. Work on the relational form in the context of information transfer would certainly be worthwhile.

Complex Data Structures

Introduction

Small files are readily searched by almost any technique. As was discussed in the section on Tabular Data Structures, small files may be stored sequentially and are easily searched sequentially or, if ordered, by rapid binary search. Simple indexing also aids retrieval. If the file is lengthy, however, a more complex structuring is useful. Consider the process which one normally goes through in looking up an item in the dictionary. Take the word "data", for example. One flips the book open at the d's, notes that "da" is at the beginning of this section, moves rapidly to "dat" and quickly arrives at "data". Notice that the search is in no way sequential through the entries, but is a branching process.

The idea that data might be best stored in such a branching - or tree - structure seems to have been first proposed in a now classic paper [Sussenguth, 1963]. A portion of a dictionary arranged in a tree structure is shown in Figure 1. The root (at the zeroth level) may be thought of as an entry to the dictionary as a whole. Following the graphical representation of Figure 1, the root might contain a list of pointers to locations containing information for each letter of the alphabet on the first level. Each of these first level locations contains the pointers to the second level, etc. Thus to find "data" one starts at the first level "d", proceeds to the node labeled "a", searches for the "t" branch, and finally arrives at "data". This brief description gives only the basic idea of a tree structure. Many variations are possible. For example, the dictionary entries themselves are likely to be stored in a linear list. The tree structure is then just an index, so that when one arrives at "data" one does not find information on the word "data", but only a pointer to the listing of "data" in the sequentially - arranged dictionary. In other situations, the entries themselves may contain the structuring, e.g. pointers to branches may be stored as part of the entry. Details of this kind are important at the programming level, but one does not need them in order to develop a good feel for the state of the art.

A tree structure can either arise in a natural way or be imposed on the data as an aid to searching. The dictionary example above illustrates a natural tree structure. Such a structure can also arise from a listing of attribute values (Cf. Introduction to Tabular Data Structures section). Suppose each record contains three attributes, A, B, and C, with each attribute taking on either of the values 1 or 2. Then to locate a record with a given set of attribute values, one might search through an index tree such as that in Figure 2. The asterisk indicates the pointer to the record with A=2, B=1, C=1. If some attributes have more values than others, the complexity of the tree will differ according to which attributes are placed on which level. A theoretical study of this problem has been published [Rotwitt and deMaine, 1971].

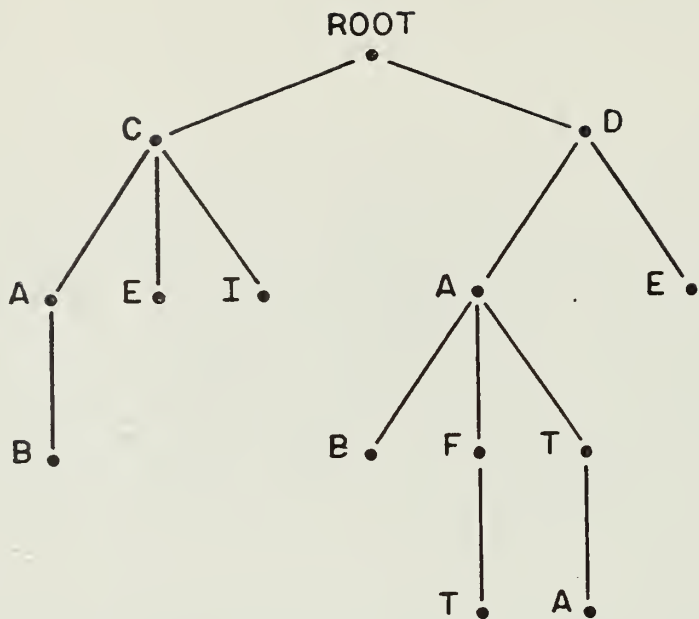


FIGURE 1

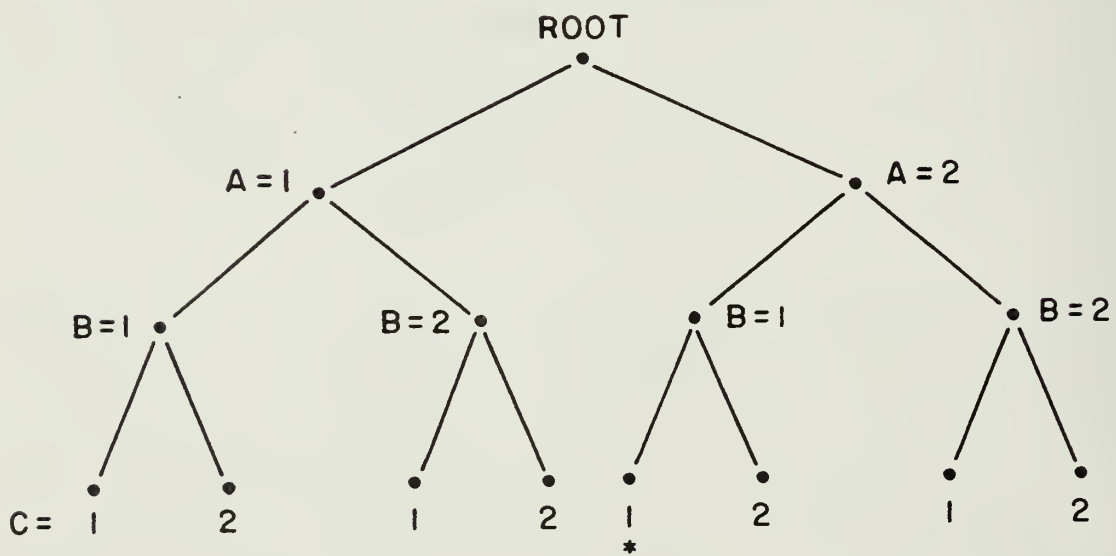


FIGURE 2

The imposition of tree structures on data to improve retrieval time has been a subject for much study ever since the appearance of Sussenguth's paper. Sussenguth analyzed a simple model and showed that if the number of branches from each node is 3.6 (on the average), the expected search time is optimized. Sussenguth also introduced the very useful doubly-chained tree, in which links not only join one level to the next, but also connect horizontal sets. The doubly-linked version of the tree in Figure 1 is shown in Figure 3. Such horizontal links facilitate retrieval. It should be noted that, although complexity in structure and linkages decreases retrieval time, it tends to complicate updates and additions to the file. It is for this reason that recent work tends to emphasize the overall behavior of data management systems and is not so concerned with optimization from a single point of view.

The discussion above has assumed that records, as single entities, are related to one another hierarchically. It is also possible for the individual records themselves to be structured. Consider a personnel file. There is probably a field with information on whether the person is married or unmarried. If the datum is "married", there may be links to a lower level on which family members are listed, and from these there may be links to data (age, insurance coverage, etc.) on these family members.

Finally, structures more complex than simple trees are appropriate for certain data. For example, suppose one wishes to store a model of a computer network in order to simulate a network routing algorithm. It is reasonable that information about each node (stored in one or more records) should be logically linked in the same way the network nodes are linked. This would allow one to readily follow simulated network traffic. Such general networks of information are useful for various special purposes. They do, however, have the disadvantage that they are not as amenable to systematic search procedures as are simple trees.

Recent Research

Optimal Trees. Several workers have followed up on Sussenguth's early analysis of tree structures. Patt [1969], for example, noted that Sussenguth's analysis assumed that all data are found at the same level of the tree. He reworks the problem assuming variable-length paths and gets two types of result: optimal search strategies for given trees and optimal tree designs in case only the number of terminal nodes (i.e., records) is specified. A recent paper [Stanfel, 1973] generalizes the model still further. In computing search costs, Stanfel allows for the possibility that horizontal steps through a doubly-chained tree may be more costly than vertical steps. His model is sufficiently complex that the optimal tree must be found by the tedious process of integer programming.

Organization of Structure Types. Early papers on data structuring generally contained extensive discussions of particular schemes. There was no overview or organization of structure types to provide the data base designer with some perspective and aid him with detailed structure choices. As a recent paper [Senko et al., 1973] describes the situation, "the state of the art of information structuring contains a complex of ill-defined observations that are far from being reduced to the precision we desire." Senko and his co-workers therefore "seek to construct a complete, faithful

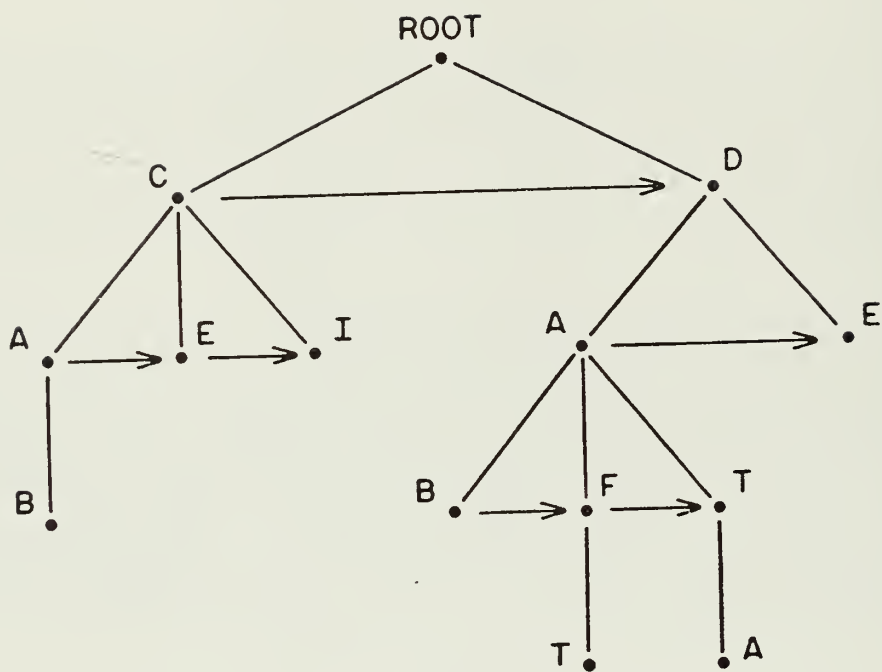


FIGURE 3

description of the intrinsic characteristics of structured information." They feel that they have still only provided a fairly intuitive approach and hope that it "makes useful common sense now and will yield to better definition in the long term."

Actually, at least two earlier papers appeared which also attempted to lay the groundwork for a better understanding of data structures. One of these [Hsaio, 1971] is primarily concerned with record organization. Hsaio's model allows for a complex (network or hierarchical) structuring of each record. He assumes that all records in a file are structured identically, so that information about this structure needs to be stored only once. Furthermore, he identifies a set of parameters which serve to locate each field within the structure. Like Senko et al., Hsaio says that his paper "is intended to be intuitive." At about the same time a paper [Earley, 1971] appeared which uses graphs to represent data structures. Earley's model is particularly interesting in that access paths are an intrinsic part of it. He also presents "the beginnings of a mathematical formalism" for the model, but remarks that "considerably more work is required to complete the formalism."

Modeling and Evaluation. Many of the early papers on data structures were largely descriptive. The merits of various structures for various applications were hotly argued, but there was no valid basis for settling these arguments. The overviews and attempts to organize the subject of data structuring (discussed in the paragraphs above) were of some help. The next step was to take these systematic views of data structures and use them to design a simulation model for a general data management system. Such a model can encompass various structures and access methods. Given information on the data and on usage patterns, one can, for example, use the model to carry out simulation studies and determine a good (if not optimal) structure before building a data base.

Indeed, Senko's attempt to more precisely define data structures was as a preliminary to the development of a "complete model for the representing, storing, and retrieving of structured information." A previous, less general model [Senko et al., 1969] was used successfully in the simulation of the overall behavior of management inquiry systems. The simulation program was provided with parameters describing logical and physical file layouts, search strategies, file content, and query types. It then automatically generated statistics on search time. A good introduction to this simulation system is contained in a general survey of information system modeling [Senko, 1972].

Cardenas [1973] notes that the modeling work of Senko and his colleagues at IBM concerns itself with many details specific to IBM systems. He therefore proposes a "more macroscopic-based model," one for which "a knowledge of file structure details and programming details is not needed." The general approach is, of course, the same. The simulation system is given parameters characterizing the data base, query complexity, and the storage devices. The system then estimates storage costs and average retrieval times. Cardenas summarizes a number of test results on real data bases. The structures compared were of the doubly-chained tree, inverted file, and multilist types. (The latter two are discussed in the Tabular Data Structures section of this report.)

Mullin [1972] describes a file system simulation program based on Earley's graph model of data structures and access paths. He presents a fairly comprehensive language for specifying both data linkages and access mechanisms. An interesting feature of the simulation system is a "data request generator" which churns out random queries to drive the simulator.

The most recent work in information system modeling [Frank and Yamaguchi, 1974] is still in the preliminary plan stage. This work grew out of a University of Michigan data translation project, and data translation between different hardware/software systems is emphasized as an important tool for system comparisons.

Winkler [1971] has proposed an entirely different methodology for the selection of an appropriate file structure in a specific situation. His idea is to develop an equation for retrieval time in terms of parameters describing data structures, retrieval algorithms, average properties of queries, etc. He then, by straightforward computation, obtains insights into the dependence of retrieval time on the various parameters. A similar approach has been taken by Gottlieb and Tompa [1974]. They describe "an algorithm for choosing suitable data linking methods from a class of available storage schemata." The heart of their algorithm, however, is a simulator which counts number of operations required by a particular program in a particular environment. These counts are then inserted in analytic formulas for further evaluation.

Automatic Structuring. As we remarked earlier in this report, the increasing size of data bases makes it imperative that techniques be developed to involve the computer more closely in the data structuring problem. Before this can be done, algorithms must be designed for structuring given data without human intervention. As we mentioned above, certain data (e.g. an alphabetical index for a dictionary) fall naturally into a tree structure. Other data have no obvious hierarchical structure. If a tree structure is desired to improve access, a method for imposing it must be devised.

An early approach [Arora and Dent, 1969] to automatic generation of a tree structure assumed ordered data which are entered into the system randomly (i.e., out of their natural order.) The first datum to arrive becomes the root of the tree. The second is stored at the next level, either to the right or to the left, depending upon whether it is greater than or less than the root. For example, if the first two items are numbered 5 and 3, 5 is the root and 3 is stored in its left branch node. If the next item is number 8, this is stored in the right branch down from 5. A binary tree (at most two branches from each node) is generated by this approach. Figure 4 shows the tree generated from the sequence 5, 3, 8, 4, 2, 10. If the data

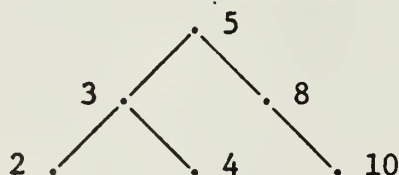


Figure 4

are entered truly randomly, the tree is fairly well balanced (i.e., does not consist of a few very long strings) and is rapidly searched. The problem with this scheme is that each datum (or record) must be assigned a number. Coffman and Eve [1970] suggest that this be done by a simple algebraic transformation of the record key or name (which may not itself be appropriate for locating the record in a hierarchy). Recently Huang [1973] has suggested synthesizing a binary tree structure for the data from user-specified binary relations between the key values. This approach includes as special cases the methods proposed by Coffman and Eve, and by Arora and Dent.

An interesting generalization of the automatically generated binary tree has appeared within the past few months [Finkel and Bentley, 1974]. Pointing out that the binary tree is only good for structuring on the basis of one key value (i.e., in one dimension), Finkel and Bentley introduce the "quad tree" to extend the idea to two keys (or two dimensions). Each node in this case may have four branches, each one corresponding to one of the four surrounding quadrants in two-dimensional space. Extensions to higher dimensions are straightforward.

An entirely different approach to automatic structuring has recently been proposed [Casey, 1973 (b)]. Casey assumes that a listing of all records retrieved by each of a set of past queries is available. Assuming that future queries will be similar, Casey proceeds to place the records in a tree structure which is optimal in the sense of minimizing total length of search for the given set of queries. "Total length of search" is defined to be the number of tests performed in answering all the queries. Casey discusses his grouping algorithm in detail and presents some experimental results. The scheme is undoubtedly too complex and time-consuming for large files. Casey suggests, however, that the same query data be used for pre-clustering the records, so that each tree node then may correspond to a whole cluster of records (Cf. Clustering and Partitioning section of this report.)

Summary and Assessment

Although a large amount of work has been done on complex data structures, there still appear to be open problems. For example, those researchers who have attempted to bring system and order to the subject have all stated that much more needs to be done. It is questionable whether any fresh approach to the problem would be noticeably more successful. Perhaps more work should be done along one of the lines already proposed. But if so, it is not obvious which line is the one to follow.

The modeling of overall data management systems is still a useful endeavor. The most elaborate modeling systems of the past have been the result of large-scale efforts at IBM, which has the resources to continue to be the pace-setter in this area. Evaluation of file structures by developing analytic expressions for retrieval times is an attractive idea, but one which it may not be possible to push very far.

Casey's approach to automatic structuring has some very nice features, especially the basing of the structure on actual query data. Although in its present form Casey's method is inappropriate for files (or data bases) of any size, perhaps some simpler variation can be devised which will be of more practical use.

Hashing

Introduction

In the early days of data processing, files tended to be small and access needs limited. It was convenient to store a set of records consecutively in memory and process the file by accessing each record in turn. For example, in the case of a personnel file used mainly to generate periodic payrolls, such techniques are completely adequate. If one wants to retrieve individual records at random, however, a linear search of the entire file can become very time consuming, particularly as the file becomes large.

An early solution to this problem was to introduce indexing, which is discussed in more detail elsewhere in this report. Indexing is a very natural idea and completely analogous to the notion of indexing a book. Consider again the example of a personnel file. If the record for each person is fairly lengthy the file is likely to extend over, say, several tapes, and direct search of the file requires much changing of tapes. It is therefore advantageous to first search an index which gives name vs. record location, so that the precise tape and location on that tape are pinpointed for rapid retrieval of the desired information. The problem is that the index itself may grow very large. The index then becomes costly to search and also takes up a great deal of memory space which might better be utilized for real data.

Another approach has been to sort the file; a personnel file might logically be arranged in alphabetical order. Such an ordered file is readily searched, just as one finds it easy to locate a number in the telephone directory. The problem with this arrangement is that it is difficult to insert new records into the file. If the records are physically stored in consecutive locations, insertion of a record requires physically moving all following (or preceding) records.

For data stored on large, random-access storage devices, the indexing approach was the preferred solution. There was no particular advantage to storing records in consecutive locations, while the difficulty of record insertion provided a strong impetus in the other direction. It then became important to determine ways of making the indexing process more efficient. Hashing (also known as "key transformation" or "scatter storage") is a method of doing away with the index altogether, while retaining its function.

The idea behind hashing is that, since the index simply describes a mapping from record identifiers (names or other unique keys) to memory locations, this mapping could just as well be defined by an algebraic function. In this way the storage and searching of a large table are

replaced by a rapid, direct computation of the desired address. This clever idea originated very early in data processing. (A good review of early techniques is available [Morris, 1968]). However, details of implementation (e.g. the algebraic function to be used) have a more severe effect on the usefulness of the scheme than one might think at first glance. In the next section we shall discuss the problems that arose as well as recent work on solutions.

Recent Research

First, let us briefly consider how a hashing scheme for a personnel file might work. Suppose that the key to be transformed into an address is a person's name, which can be looked at for algebraic manipulation purposes as its computer encoding into a string of binary bits. Denote this key by k . Now the set of all possible person names is enormous compared to any memory size, so the possibility of using a one-to-one transformation may be eliminated. One therefore keeps the function simple (for rapid computation) and remembers that the numerical result must be less than the memory size (to obtain a valid address). For example, one simple method is to divide k (interpreted as a binary integer) by the memory size (or related integer) and use the remainder as the hash address. Other popular schemes are based on various bit-string manipulations [Page and Wilson, 1973, pp. 164-5].

The basic problem of hashing is that of "collisions", or the mapping of more than one key into the same address. A number of transformations do a reasonable job of reducing the number of collisions to an acceptable level, so that devising new transformations is no longer a useful endeavor. In any particular situation, however, the choice of hashing transformation may be important. A method that works well for alphabetic keys may deteriorate if extended to alphanumeric keys. And short keys are likely to hash differently from long keys. In order to provide some guidance on this question, Lum and co-workers [Lum et al. 1971, 1972] have carried out a lengthy study of eight popular hashing schemes. They applied each scheme to each of eight real files chosen to typify diverse applications. A complete discussion of their approach would require introduction of more terminology than is appropriate for this brief review. In essence, their measure of performance is closely related to the number of collisions. An interesting discovery was that the simple division scheme (described above) performs best in a wide variety of situations.

More recently, Lum has developed a theoretical approach to analyzing key transformation performance [Lum, 1973]. The idea is to define a "space" of keys and to study the abstract properties of the hashing function as a transformation on that space. Using this approach, Lum is able to provide a theoretical basis for the empirical results he reported in earlier papers. The approach seems to have potential for further development and application to the understanding of hashing transforms.

Given that collisions are bound to occur, the next problem is what to do when they do occur.

Originally collisions were handled by storing the set of records whose keys hashed to the same address in a set of consecutive locations beginning at that address. Thus both record insertion and retrieval involved (and still do involve) computation of a hash address plus a subsequent serial search (for the first empty spot in the case of insertion and for the desired record in the case of retrieval).

A difficulty which immediately arose was the collision of these record sets. That is, some hash addresses might be sufficiently close together that there might not be room in between them for the complete set of records hashed to one address. As a first approach to a solution, an attempt was made to distribute the hash addresses fairly uniformly through the memory, and most hash functions are reasonably successful at this. This technique does not completely solve the problem, however, and more sophisticated ideas have been developed.

The earliest such scheme was a straightforward generalization of consecutive storage known as "linear hashing." In this scheme, if the computed address h is occupied, then the addresses to be subsequently tried are $h + a$, $h + 2a$, $h + 3a$, . . . , where a is a given integer. The addition is of course modulo the table size. Note that setting $a = 1$ gives consecutive storage. The idea is that it may be unlikely that $h + na$ (n an integer) will be a hash address, so that various hash sets will store in intermingled fashion and set collision will be postponed. In practice, however, clustering is severe and set collisions become common. Set collisions do not, of course, destroy the scheme, but do lengthen search paths considerably.

The next obvious extension was to "quadratic hashing", in which addresses $h + na + n^2b$ ($n = 0, 1, 2, \dots$) are searched in turn. Maurer [1968] introduced this method and within a few years several papers by various authors elaborated on and analyzed the method. (See, for example, [Hopgood and Davenport, 1972].) It appeared that although quadratic hashing eliminates the clustering of linear hashing, it introduces a secondary clustering which is nearly as troublesome. A modification of quadratic hashing known as the "quadratic quotient method" [Bell, 1970] eliminates this secondary clustering. The trick is to make the parameter b vary with k . To be precise, Bell assumes that the hash address h is computed by the division method and suggests using the integer quotient itself as b (so no extra cost is incurred).

In case divisions are costly, $h(k)$ may be best obtained from simple bit-string manipulations, and some other method for varying b must be found. Within the last few months a paper has appeared suggesting that two functions $h(k)$ and $g(k)$ be defined by bit-string manipulation (e.g. from different subsets of the entire key string) and that both a and b be made to depend upon g [Nishihara and Hagiwara, 1974]. The claim is made (backed by simulations) that such a method will effectively eliminate secondary clustering.

A completely different and very promising line of research has appeared within the past year. A drawback of hashing has been that only the one selected record identifier has been used to compute the hash address. This makes it difficult to do any querying of the file other than simple retrieval of a specified record. Retrieval on the basis of multiple key values has been handled by inverted file and complex indexing techniques. A recent paper [Rothnie and Lozano, 1974] proposes combining inverted file techniques with multiple-key-hashing (i.e. transformation of some n -tuple of keys, $n \geq 1$). The choice of the particular combination to be used is optimized on the basis of a cost analysis which includes information on the data and access traffic. Optimization is carried out by heuristic methods, but Rothnie and Lozano report on empirical studies that show that the performance of the suboptimal combination obtained is very nearly best and substantially better than that of inverted files alone.

Summary and Assessment

Although hashing is an old and well-studied area in data handling, there is still room for future work. Basic techniques seem to be as fully developed as is needed for practical use. The main problem - that of collisions - seems to be satisfactorily solved. Nevertheless two recently introduced lines of research seem to be very promising. One, the mathematical analysis of hashing transforms [Lum, 1973], could probably be pursued in various directions (e.g. one could introduce different performance measures) to obtain further insights into which hashing function should be chosen in a particular situation. The other new line of research is multiple-key-hashing and its combination with other access methods [Rothnie and Lozano, 1974]. Possible extensions of this work are numerous. For example, Rothnie and Lozano carry out their analysis for a paged memory environment. Other assumptions might change the results substantially. Rothnie and Lozano themselves state that the most important area for future research is "the degradation of retrieval performance as file parameters change over a period of time", and the associated development of a dynamic reorganization strategy. In summary, it appears that there is still useful work to be done on the subject of hashing, and particularly on its use in conjunction with other accessing methods to improve overall system performance.

Clustering and Partitioning

Introduction

As data bases grow in size, it becomes increasingly important to group records and files for efficient retrieval. Small file systems which fit into primary memory can be readily searched by any sequential or indexing method. But as soon as the files grow so large that they must be stored on secondary memory devices (e.g. tapes) and transferred to main memory one block at a time for searching, it becomes imperative that one give careful thought to the choice of record sets to be stored as such a block. The terms "clustering" and "partitioning" both describe this basic process of blocking the records. Only the viewpoints are slightly different - "clustering" being from the point of view of grouping together the individual records and "partitioning" being from the point of view of subdividing the whole data base.

There have been many lines of research in the area since its inception only a few years ago. Different applications have led to different approaches based on different needs. Even though much work still needs to be done, research in the area is already sufficiently extensive that this report should be considered only an overview of some of the more interesting lines of work.

Recent Research

Blocking According to Access Frequency. One of the early interesting papers suggests organizing the data simply on the basis of access frequency, without regard to any inherent interconnections between the data items [Ramamoorthy and Chin, 1971]. The assumption is made that only a single record is being searched for and that the file is ordered so that a binary search for the desired record key can be carried out. (For an explanation of the binary search technique, see the Introduction to the Tabular Data Structures section of this report.) For a large file, even a binary search may be very inefficient, especially if many items are seldom retrieved. Ramamoorthy and Chin propose that statistics be collected on how many times each record is retrieved and the records then be organized into blocks of approximately equal retrieval frequency. Thus block 1 would contain the group of most frequently retrieved records, block 2 a group which is less frequently retrieved, etc. The size of each block is set to be $2^j - 1$, where j is some integer. This choice of block size allows efficient binary searching of the block. The particular choice of j depends primarily upon the observed access-frequency groupings. For example, if it is observed that 50 records are each retrieved 100 or more times a day, while no other record is retrieved more than 25 times, block 1 might consist of 63 records ($j = 6$), the 50 most retrieved records plus 13 others retrieved about 25 times. If retrieval also involves transfer of blocks back and forth between primary and secondary memory, this fact should also be taken into account in choosing j . Once the records have been blocked, the search for a particular item proceeds by binary search of each block in succession, beginning with block 1. The probability of finding the desired record early in the search is now considerably larger than it was in the original unorganized file.

Clustering for Document Retrieval. Document retrieval poses entirely different problems than does the single record access discussed in the paragraph above. In this case one wants to retrieve a whole list of documents, all those pertaining to a certain set of key words or otherwise having some prescribed attributes. Thus one might want to retrieve all documents having to do with clustering and data management - but not all clustering papers or all data management papers. As the published literature in all areas of research proliferates, demand for automatic literature searching systems increases and research into making such systems effective is extensive. For the sake of brevity, we will here only sketch the key ideas.

An important point to notice about document retrieval is that it generally need not be exact. If some documents are retrieved that are peripheral to the interests of the person making the request, they are readily discarded. On the other hand, the failure of the system to retrieve a useful document may be somewhat disconcerting, but probably not fatal. Thus "efficiency" in document retrieval jargon does not usually refer to access time or the speed of processing requests, but to how close the match is between an "ideal" response set to a query and the actual set produced by a system. At the same time, clustering to improve speed is a necessity for large systems. These considerations lead to the clustering of similar documents, with subsequent retrieval of whole clusters in response to requests.

Van Rijsbergen and Jardine have done some of the most influential work along these lines. [Jardine and Van Rijsbergen, 1971], [Van Rijsbergen, 1971]. A single-linked hierarchical clustering method is proposed. This is a method which on the first step groups together items that are within one distance unit of each other, on the second step groups items (and groups) less than 2 units apart, and so forth, until at the last step one cluster encompasses the whole set. The clustering metric (measure of distance) is based on the number of document keys in common. Thus, documents with many key words in common will tend to be put into the same cluster. To each cluster is assigned a "representative", a vector indicating the key words associated with an "average" member of the cluster. Retrieval requests are also in the form of such a key-specification vector. Requests are therefore matched against cluster representatives and the cluster corresponding to the closest match is returned. The hierarchical structure allows clusters of various sizes and relevance to be returned. In the context of the example mentioned above, there may be one cluster for all documents on data management, another for all documents on clustering, and a third for all documents on both subjects. The third will appear at a different level in the hierarchy. The user can then specify whether he wants a small group of documents fitting his request very precisely (and perhaps missing some relevant items), or a large group of documents fitting the request more loosely (and probably containing many irrelevant items). Early tests of the scheme were carried out on a rather small system (200 documents). More recently work has been reported [Van Rijsbergen, 1974] on a more general search strategy, which allows for the retrieval of more than one cluster. Extensive tests on several document files are also discussed.

In a large and varied document system, the vectors describing requests and cluster representatives may themselves become very long and unwieldy. (They are usually binary vectors, each bit indicating the presence or absence of a particular keyword in the description.) This problem may

be solved by compression techniques (cf. section on Compression) and suggestions as to specific methods have been made [Crouch, 1973].

Finally, in recent months a quite different approach to document clustering has been proposed [Chien and Mark, 1974]. A "binary index vector" is attached to each document to describe its contents (just as for requests and cluster representatives as described above). The weight or norm of such a vector is defined to be the number of non-zero components. One proposal is that documents be simply clustered according to their weights - i.e. all documents of weight n are stored in "bucket" n . A set of theorems are developed which indicate whether or not it is worthwhile searching in bucket n for a request of a given weight. A more sophisticated variation on this approach is also discussed. Insofar as a document with a large weight is apt to be relevant to many requests and thus be accessed more frequently, this approach would seem to be closer in spirit to that of Ramamoorthy and Chin than to the common document retrieval methods.

Other Clustering Methods Based on Similarity of Keys. Rettenmayer [1972] suggests clustering an arbitrary file with similarity between keys (or attribute values) used as a measure of distance. He uses a centroid-type clustering scheme, in which items are grouped together according to which of several central points (centroids) they are closest to. This should lead to tighter clusters than the single-link method proposed by Van Rijsbergen. In addition, although Rettenmayer also compares retrieval request vectors with cluster representatives (the centroid in this case), he does not suggest that the whole cluster with the best match be returned. Instead the best matching cluster or clusters are merely identified as those most likely to repay further searching. The whole process is therefore designed to yield the exact set of records desired, and not just an approximation to that set, as is the case with document retrieval. Rettenmayer has done some testing of the method, but only on small files (up to 100 records).

A very recent paper [Skinner, 1974], written more from the point of view of artificial intelligence than from that of data retrieval, proposes a novel use of clustering techniques. The suggestion is made that clusters be formed that are very closely linked (i.e. the items have many keys in common). Assuming that items with many features in common are likely to have even more in common, Skinner uses a heuristic approach to "infer" information that is not actually contained in the data base, the inference being made from known information about other members of the same cluster. Though hardly foolproof, this approach could be a useful aid to answering important queries in situations where some data is unavoidably missing.

Clustering Based on Simultaneous Retrieval. We have seen that clustering for document retrieval, as well as for other purposes, is generally based on similarity between keys, or attribute values. The idea is that similar items tend to be retrieved together and so should be clustered together. It is a natural extension to consider the possibility of determining directly (by monitoring the queries) which items actually are retrieved together. Clustering then may be carried out on the basis of this retrieval data.

Several years ago the suggestion was made [Wong and Chiang, 1971] that records be clustered into small, disjoint sets ("atoms"), each atom

being always retrieved as an entity. Thus in this scheme clusters consist of sets of records that are always retrieved together. For certain applications this may turn out to be a reasonable approach, but (as Wong and Chiang point out) in other applications the atoms may turn out to be simply the individual records and no clustering at all has been accomplished.

Ghosh [1972] takes a similar approach. Assuming that the storage medium is one with linear accessing (e.g. tape), he suggests that all records retrieved by any one query should be stored in consecutive locations. Organization of a file in this fashion requires that the set of allowable queries be completely specified in advance. Notice that the clusters are not necessarily stored in disjoint fashion. For example, if query 1 retrieves records a, b, c and query 2 retrieves records b, c, d, then we would store a, b, c, d in consecutive locations for rapid retrieval of either set. The obvious problem that arises is that if records a, c, e are requested by a third query, these are not located consecutively. Ghosh has done some theoretical work on when a given set of queries and data do have the "consecutive retrieval property", i.e. the potential for being ordered according to his scheme. It should be noted that, although it seems unlikely that any complete query set in use would have this property, it may be that a heavily used and important subset might.

Recently a particularly promising approach for using information on queries to cluster and/or structure a data base has been proposed [Casey, 1973(b)]. Casey organizes data on queries into a table of "response patterns." Essentially, this table is a binary matrix Q with entries q_{ij} , where $q_{ij} = 1$ if the i th query retrieves record j and $q_{ij} = 0$ otherwise. Thus the j th column, giving complete data on which queries retrieve record j , is the "response pattern" for record j . Casey suggests that records be clustered with Hamming distance between response patterns used as the clustering metric. He also develops a scheme for deriving an hierarchical structure from Q . This scheme is briefly discussed in the section of this report dealing with Complex Data Structures.

A recent report from IBM [Gorenstein and Galati, 1974] also proposes use of the matrix Q (but, strangely enough, without referencing Casey's work). Two clustering (or partitioning) schemes are discussed. Euclidean distance between response patterns is used in both. Gorenstein and Galati also put some thought into the development of an optimal (or at least good) replacement algorithm for transferring the clusters (or blocks) between levels in a storage hierarchy. Limited experimentation has been carried out on a large data base (3.4 billion bytes).

Summary and Assessment

Data clustering, a relatively new area of research, will become increasingly important in the design of large information systems. The earliest extensive work in the field was done in the area of document retrieval. Most progress in document retrieval clustering, however, has had the drawback that it was based on the premise that it doesn't matter if some records are missed. This premise is usually considered untenable in most other applications. On the other hand, in cases where the data base is very large and may not have all the correct, up-to-date information anyway, use of

document retrieval techniques may be a reasonable approach to getting quick answers. If "exact" answers are desired, Rettenmayer's idea of following up cluster retrieval with a search of the cluster to find the precise information requested looks very promising. Further study and testing of this method would be worthwhile.

Perhaps the most promising line of research is the investigation of clustering based on the actual, observed query data. The recent work by Casey, and by Gorenstein and Galati, appears to be in a direction that it would be very profitable to pursue. Data bases of the future should ideally have the capability of monitoring and analyzing usage patterns, and of automatically restructuring themselves in response to changing usage. The concept of clustering data on the basis of response patterns may well be a step towards this ideal.

Introduction

Conservation of storage space has been an urgent requirement since the earliest computers. It is practically axiomatic that the amount of information that one wants to process tends to outstrip system capabilities. The availability of virtually unlimited amounts of auxiliary storage does not solve the problem, since the continual need to transfer files back and forth between memory hierarchies is extremely time-consuming. The compression of data for compact storage has therefore been a topic for thought and discussion over many years. Surprisingly, though, there has been much less formal study and attendant literature than one would expect.

We distinguish two main classes of data compression techniques. The first encompasses methods for compression of approximate numerical data, or, in the terminology of the most common applications, telemetry data. The schemes used are such things as truncation to a minimal number of significant digits, replacement of tabular data by a single entry plus a list of successive differences, etc. This report, however, is largely concerned with administrative data management. We will therefore limit our discussion to the second class of compression techniques, namely those appropriate to the economical storage of business or administrative files.

Logical Compression. One type of compaction that arose very early is so-called logical compression. The idea here is to take advantage of known redundancies, relationships between fields, known information about how many possible values a field may take on, etc. A simple example of this is the familiar designation of sex by M or F instead of Male or Female. But since the field can take on only two possible values, further compression to a single binary bit is common. A large amount of logical compression is usually built into the design of every data base. Much information is entered into the system in encoded (or abbreviated) form. Quite often, however, too much encoding on input leads to errors, so that further encoding should be left to the system. For example, M or F for sex designation may be more likely to be typed in correctly than 0 or 1. The system can readily make the conversion. Logical compression, however, always has the disadvantage that it is based on very specific information about the data.

Null/blank Suppression. Another obvious and old techniques is null/blank suppression. The idea here is that the usual administrative record contains several strings of blanks or zeros that perform no function except to space out the data. In a personnel file, for example, a 30-character field may be allotted to the person's name. A short name may then be followed by 15 or more blanks. To save space one could replace such a string by something like "b15", indicating a string of 15 blanks. In a similar way other repeated patterns of characters may be encoded. A brief and readable discussion of several such techniques is available [Ruth and Kreutzer, 1972].

Statistical Encoding. The next step after the encoding of obvious patterns was the development of a more rational basis for encoding. The idea arose that an overall saving can best be made by attaching the shortest codes to the most frequently used patterns. The group of compression techniques based on this idea is usually referred to as "statistical encoding." "Patterns" in this context are often single alphanumeric characters, although frequently appearing combinations may also be coded. Morse code is actually an example of statistical encoding, since the most commonly used letters are represented by the shortest code groups.

Most statistical encoding schemes are based on Huffman codes, a well known binary encoding which is optimum in the sense that a set of data encoded by the Huffman algorithm could not be expressed in fewer bits. (Huffman codes are discussed in many texts, for example [Page and Wilson, 1973, pp. 22-4].) Huffman codes also are desirable because they have the "prefix property"; that is, no bit string representing a character is the initial segment of the bit string for some other character. If a code has the prefix property, no spacers are needed between characters. Although Huffman codes have been in existence for a long time [Huffman, 1952], the first suggestion that they be used for file compression was put forth only six years ago [Maurer, 1969].

Recent Research

Evaluation and Experimentation. It has been generally accepted that Huffman encoding - or some very similar variable-length statistical encoding scheme - is the best approach to compressing an arbitrary file (i.e. one in which logical compression has already been carried out and no more information useful to special-case compression is known). Everyone of course agrees that a reasonable amount of logical compression should be the first step in saving storage. Some of the more interesting of the recent papers deal not with the devising of new schemes, but with experimentation on the old. One such paper describes compression carried out on a large commercial data base [Mulford and Ridall, 1971]. A combination of five compression schemes was used and a compression ratio of 4.7 was obtained. Since the devices used included elimination of obvious redundancies, it is difficult to evaluate this result. (It is easy to imagine a data base inflated with absurd redundancies.) A more useful result was obtained by Ruth and Kreutzer [1972]. They tried Huffman coding on a large Navy inventory file (over a million COBOL records) and obtained a compression ratio of 2.58. Furthermore, their analysis showed that compression had effected a net cost savings of \$10,000 a month.

Suboptimal Codes. Feeling that Huffman coding may not be best from the point of view of implementation, a group at IBM began work in 1970 on the design of "suboptimal" codes. The object was to design codes that would bring "substantial gains in space, speed, or price of special hardware for implementing the encoding and decoding operations," but without significant loss in compaction or the weakening of the prefix property. For example, they introduce the idea of a copy code, a prefix corresponding to the Huffman encoding of a block of most infrequently used characters. But the prefix, instead of being followed by further code bits, indicates that a copy of the character is to follow. By this device, entries for infrequently used characters may be omitted from the encoding/decoding table. Another

scheme for speeding up the decoding process is to attach to each code word a prefix indicating the length of the word to follow. Much of this work, including discussion of hardware and software implementations, has up to now only been available in the form of United States Patents. Recently, however, IBM has published a report which discusses the work and gives examples and experimental results [Mommens and Raviv, 1974].

Automatic Analysis and Compression. In nearly all the work on data compression, it has been taken for granted that a considerable amount of manual labor goes into the process. This labor takes the form of selecting items for logical compression and choosing the set of characters (and patterns) for Huffman coding, as well as in most cases developing the statistical data. Even the encoding and decoding tables are often set up a priori and with some programmer work. Clearly it would be more efficient to utilize the power of the computer to aid in these tasks.

An important step towards automating compression has recently been taken [McCarthy, 1974]. McCarthy has developed a software system which statistically analyzes a sample of the file to be compressed and then chooses the set of character strings to be encoded (including the machine's character set). The system then uses the Huffman algorithm to generate optimal codes and, finally, encodes the files. The whole procedure is said to be completely automatic and machine independent.

Summary and Assessment

The trend towards more systematic and automatic encoding procedures is an important one and will doubtless continue. Logical compression is too much dependent on the properties of a particular file and on manual labor to effect as much compaction as is desirable.

There is one difficulty with pushing automated compression too far; namely, it makes decoding a prerequisite to answering queries. Although most compression schemes assume that decoding is a necessary part of searching the data base, it has been recently pointed out [Alsberg, 1975] that considerable savings could be made by compressing queries instead and then searching the compressed data base. The problem with this is that important information, such as the ordering of certain field values, may be lost in the compression. For example, a query may ask a personnel file for the names of all employees over age 60. A mindless, automatic compression scheme may have not only destroyed the normal ordering of the entries in the age field but also may have destroyed the individuality of the age field itself. Alsberg suggests compression of each field separately, and by an encoding technique which carefully maintains desired features, such as orderings.

It may be that the primary applications of completely automated compression systems, such as McCarthy's, will lie in the areas of file transmission over a network and of storage of large, seldom-accessed files (e.g. back-ups). The possibility of implementing automatic compression for these applications is worth studying; the potential savings are enormous.

On the other hand, in the case of large data bases that are continually being hit by complex queries, some combination of compression techniques is probably best. Work still needs to be done on devising compression strategies which optimize overall system efficiency.

Introduction

In this section we will briefly consider languages which handle data in some sense. At the lowest level, there may be a so-called storage-structure language with which the physical structure of the data is specified. The storage-structure language is concerned with such matters as specific addresses and the particular encoding of values. The usual so-called data definition (or description) language (DDL) is more concerned with the definition of logical structure (i.e., hierarchies, lists, etc.). The DDL may be thought of as providing an extension to the familiar FORTRAN or COBOL formats, in the sense that it allows the user to format his data in complex ways. That is, in addition to simple one and two dimensional arrays of data, or field-by-field specifications of a record as a simple list, the DDL allows for hierarchical and network structures. The DDL may be used at various levels. The Data Base Administrator may use the DDL to establish the logical structure of the entire data base. At the same time, an individual applications programmer may establish his own (slightly different) view of some portion of the data through use of the DDL. A fairly detailed specification for a DDL was drawn up by the CODASYL Data Base Task Group [1971]. The aim of this group was in large part the enhancement of the data structure capabilities of programming languages (i.e., the format extension mentioned above). Other DDL work has been more concerned with generating standards to facilitate data transfer between different hardware/software systems.

In addition to the DDL, there is usually, at the next level up, a Data Manipulation Language (DML) [CODASYL Data Base Task Group, 1971]. To quote the CODASYL Report, the "DML is the language which the programmer uses to cause data to be transferred between his program and the database." It is usually not a complete language in itself but relies on a host language to provide a framework for it. Procedures for handling the data are then written in a mixture of the host language and the DML.

The DML provides for elementary data manipulation and may be adequate in an environment where only simple operations are to be performed. For example, processing a payroll is a routine operation which can be accomplished by an applications program which uses the DML. But suppose one wants to ask random questions of the data base (e.g., which people earn more than \$10,000?). It is impractical to ask that applications programs to handle every such question be written and called as needed. What one needs in this case is a convenient user interface, a so-called query language. Such a language is an integral part of any interactive information system and may be designed for varying degrees of user convenience. In this report we have somewhat arbitrarily subdivided our discussion of query languages into subsections on algebraic languages, structured-English languages, and natural languages, in order of increasing user convenience. Of course, the more structured or algebraic the language is, the less work the system must do to interpret the queries. As in most data management problems, there is a balance to be struck.

In the following paragraphs we will discuss some of the recent thinking on data languages. The reader should be aware that we are not attempting to cover the field anywhere near completely in this brief report. Much experimentation is currently going on in language development at all levels. Many special purpose data bases have their own special purpose query languages (as well as lower level facilities). Information on many such languages is not available in the literature. Discussing any language (or class of languages) in any detail is far beyond the scope of this report. We must content ourselves here with simply noting some examples of recent work and indicating apparent trends.

Some Recent Work

Data Definition Languages. As we mentioned in the introduction, there may be two levels of data definition - the logical and the physical. Sibley and Taylor [1973] have published a good overview emphasizing the need for the lower level, physical, structure definition. They point out that the translation of data between systems - or the setting up of a distributed data base on a network - requires a language for defining storage structures precisely. They note that "if a language can be developed to describe storage formats in sufficient detail, statements in the language can then be used as input to programs which would reorganize a file or transform the file into a format suitable for use by other hardware or software systems." Sibley and Taylor urge that a structure definition capability be integrated into a general data definition language which includes logical specifications as well.

On the other hand, Dennis [1970] earlier argued "that the general problem of data exchange is no less difficult than the problem of program exchange, and that the concept of a 'data description language' is not a solution to the problem." Nevertheless, Dennis feels that a DDL is a necessary aid to exchange. He proposes that a suitable class of abstract data structures be used as a common language for data interchange and that the DDL be used to formally define this class. The DDL might then also enter into the process of translation back and forth between the intermediate structures and the original or target structures.

Data Manipulation Languages. As we mentioned above, the CODASYL Group Report suggested a modular structure for data-handling languages. To some extent there has been a trend away from modularity. One example of this trend is the suggestion of Sibley and Taylor that a lower level physical-structure specification language be integrated into the DDL. In the other direction, a relatively recent IBM report [Boyce and Chamberlin, 1973] suggests that the data definition facility be integrated into the data manipulation language. They work in the context of relational forms (cf. Tabular Data Structures section) and propose a data definition facility "which contains a query language." They also "show how this enables the data management system to assume responsibility for data description, multiple views of data, consistency and integrity, authorization, etc." The query language referred to here is the structured-English language SEQUEL (to be discussed later) which is sufficiently structured and closely enough related to the data format that it can double as a data manipulation language. There is in fact no clear-cut distinction between a DML and a

query language. Both manipulate the data, and any query language (except, perhaps, a natural language) can have a higher-level language built on top of it, so that it is reduced to DML status.

Query Languages: Algebraic. Just as it is difficult to classify the levels of data handling languages, it is difficult to say just when a language is "structured" and when it is "algebraic." If the queries are written largely in symbols with a very rigid logical (or set-theoretic) structure, the language is clearly algebraic. But as more English words are introduced into the language, and the queries become more intelligible to the uninitiated, at some point the language becomes not algebraic, but structured English.

From the user's point of view, a query may be thought of as a request that certain pieces of information be retrieved from the data base. Thus one might ask a personnel file for the names of all persons who are female and earning less than \$10,000 per year. A natural English form of the query would be more or less as stated. A structured English form for the query might look like this: FIND employees such that SEX = F and SALARY < 10,000. A more algebraic version would perhaps use set-theoretic notation throughout and read: FIND { $x \in E$: SEX = F and SAL < 10,000}, where the brackets denote a set, and " $x \in E$:" reads "x is an element of E (the set of employees) such that". One sees that the difference between an algebraic and a structured-English language is conceptually very slight. Indeed, some workers have been led to conclude that research on retrieval languages is not very constructive. As Schwartz remarked in a symposium paper [Schwartz, 1972]: "Data base problems are generally quite simple from the logical point of view, and easily formulated in set-theoretical terms. . . . They generally only require that certain straightforward combinations of the basic operations subset extraction, union, intersection, counting, totalling, and maximization be carried out." Thus, for example, the query above could be answered by finding the subset of E with SEX = F, followed by finding the subset of E with SALARY = 10,000, and finally taking the intersection of these two subsets. The actual retrieval algorithm could be quite different from this conceptual description. One might perhaps search linearly through the file, checking each record in turn to see if it satisfies both requirements. The point is that a logical, set-theoretic language is adequate for expressing most queries.

We mention one example of a purely algebraic query language - the language ALPHA which Codd developed for use with relational forms [Codd, 1972]. A glance at Codd's paper will convince the reader that algebraic languages may indeed be inconvenient and difficult to learn to use and understand. Codd devised this language not for use, however, but in the context of an "attempt to provide a theoretical basis which may be used to determine how complete a selection capability is provided in a proposed data sublanguage."

Query Languages: Structured English. Most query languages designed with user convenience in mind fall into the category of structured English. As an example we will briefly consider the language SEQUEL ([Chamberlin and Boyce, 1974], [Astrahan and Chamberlin, 1974]). SEQUEL was briefly mentioned earlier in this report in the context of relational forms. (See the section

on Tabular Data Structures.) The query used above as an example would be written in SEQUEL as

```
SELECT      NAME
FROM        EMPLOYEES
WHERE       SEX = F
AND         SAL < 10,000.
```

To see how this works, imagine the data stored as a table (or relation) called EMPLOYEES, with (among others) columns labeled NAME, SEX, and SAL. One is therefore selecting certain items in the NAME column from the table of EMPLOYEES, the items to be selected being those for which the corresponding information in the SEX and SAL columns satisfy the WHERE clause. Basic query-blocks of this form may be nested to generate more complex queries, and a combination of nesting with the ability to use a query-block to define a variable allows for considerable query flexibility. Facilities for standard operations like counting and averaging are also included.

Clearly, one can learn rather quickly how to use the basic query-block structure to extract desired information from tabular files. Equally clearly, the language is very rigidly structured and the user must have full knowledge of his data - i.e., which tables store what information and the labels attached to all tables and columns in those tables. A major effort is currently in progress at IBM (San Jose) to extend and improve SEQUEL. This effort should eventually lead to the building of a more convenient language on top of SEQUEL [Traiger, 1975].

Actually it is the rigidity of the syntax and not that of the naming that characterizes a structured-English language. It is easy enough to include a dictionary of synonyms which interprets non-standard names for items. It is also possible to include a wider range of query flexibility and variability than exists in SEQUEL. Queries written in a structured-English language may in fact be completely readable and look very much like natural English. The difference is that a query written in structured English must have followed the prescribed rules for syntactic construction, while an ideal natural language processor is supposed to be able to handle any question the user cares to ask.

Query Languages: Natural. If there actually existed processors which could understand arbitrary English queries, one would have no need to discuss natural query languages as a special class. The fact is that so-called natural languages always involve only a subset of English and some care must be taken not to present the system with a query that is beyond its interpretation capabilities. In essence, however, the language used is English; hence it is the processor techniques that are of most interest and not the language itself. One must therefore talk about the whole system.

One of the earliest experiments in natural language querying was the system REL, designed by a group of workers at the California Institute of Technology ([Thompson et al., 1969], [Dostert and Thompson, 1971]). A language is defined to the system in terms of its (1) terminal symbols and parts of speech, (2) rules of grammar, and (3) semantic routines (which determine the meaning of a phrase from the meanings of its constituent parts.)

As a sentence is read into the system, it is scanned by the parser, which builds a parsing graph based on its recognition of parts of speech and rules of grammar. Finally, a semantic analysis is carried out and the sentence is interpreted.

Any natural language contains ambiguity. The designers of REL have attacked the ambiguity problem in two ways. First, the data base itself affects the possible interpretation of a query; e.g., if by one interpretation a query is answerable and by another it is not, the first is chosen. Second, the dictionary entry for each word contains not only part of speech and definition, but also a list of "features", additional rules which aid in syntactical analysis.

Several other experimental natural language query systems have been developed. They all necessarily involve similar schemes for syntactic and semantic analysis. All are also faced with the basically insoluble problem of ambiguity. It is unfortunately beyond the scope of this report to look closer into the various techniques for natural language processing.

Summary and Assessment

The design of languages for handling data is a large and complex subject, with much research activity currently being carried on. The arguments over what approach is best for data transfer is particularly relevant to setting up a distributed data base in a network. One surely needs a flexible, powerful data definition language with the capability of defining storage structures as well. But if Dennis is correct, such a language will not suffice to solve the problem in general.

The design of natural language query systems is the subject of much current research. To a large extent, this subject belongs to the domain of the expert in linguistics and artificial intelligence. From the point of view of data base retrieval, there is a question as to how far it is worthwhile to push this effort. Indeed, a recent paper [Chai, 1974] "presents a number of arguments against using English as the query language for information management systems." Chai's contention is that the user should (and usually wants to) express his query precisely and unambiguously. For this purpose a good, structured-English query language - perhaps tailored for the particular data base - is probably optimal.

Integrity

Introduction

The term "integrity" refers to the correctness and completeness of the information stored in the data base. Bad or missing data may arise in a number of ways. Data may be typed in wrong. Communication line, hardware or software failures may cause deterioration or loss of items. Unsecured data may be either accidentally or purposefully altered or destroyed by unauthorized users. Concurrent updates by independent (but authorized) users may have unexpected effects. It has been imperative that ways be devised to maintain integrity insofar as it is possible. The devices - back-ups, consistency checks, etc. - are not completely foolproof and sometimes, as we shall see, have led to further problems.

Recent Work

Consistency Checks (data cleaning). One way to try to identify bad data is through consistency checks. Typographical errors may often be caught on input if the system has some information on what the item should be like - e.g. alphabetic or numeric, or, if numeric, within a certain range of values, etc. We noted earlier in this report (cf. Tabular Data Structures) that it may be particularly convenient to attach this sort of checking information to relational forms - and that there is an ongoing effort at IBM Research (San Jose) to develop this technique as far as possible. If these same sorts of checks are carried out periodically, they may also serve to catch hardware/software errors, as well as malicious attacks on the data.

Consistency checks may also involve combinations of data items. For example, in a personnel file, salaries may be checked by maintaining a separate total of all salaries. As another example, it may be known that each department can have only one manager; a check catches the fact that two names are flagged as being the manager. Similar types of information checks are readily devised for any particular context.

The relational data base nearly always contains built-in redundancies that may be used for checking. The process of converting data which is naturally hierarchical into normalized relational form inevitably leads to the construction of tables whose information content overlaps. These interconnections between the various tables are readily formalized and exploited for consistency checking.

In any data base organization there are bound to be items which are implied by combinations of other items. Such logical relations may also be used for checking, although it may be extremely difficult to identify them and choose some subset of logical interconnections that is most appropriate for checking.

There is very little in the literature on consistency checks. However, one recent paper [Florentin, 1974] discusses consistency auditing in some detail. Florentin proposes the use of logical notation to express constraints on data items. He feels that this approach "is a valuable aid to examining the nature of the constraints. In particular, it is found in practice that various alternative ways of expressing the constraint become apparent, and this is valuable

in providing more insight." Florentin also remarks that use of a good, consistent mathematical notation for checks allows a systematic determination of the relative costs of checking. Hence one may choose to implement a particularly cost-effective subset of all possible validity checks.

Maintenance of Backups. In case a data base experiences serious loss or deterioration of data, it is important to have backup copies of files available. These copies then can be used to restore the data base to its last valid state. The basic approach is to periodically dump the data base onto auxiliary storage (e.g., tape). Several generations of these dumps may be kept, as well as associated files containing all transactions that modified the data base posterior to the dump. Vold and Sjogren [1973] have developed a stochastic model for this dumping process. They make "two important assumptions: the first that an error is discovered the instant it occurs, the second that the dumps are all good in the sense that if an error occurs we can reload from the last dump." They develop an expression for total backup and recovery cost in terms of various system parameters and study the problem of finding that time-spacing between dumps which minimizes cost. Extensions to the model were said to be planned.

In a network, additional problems in maintaining backups arise. Catastrophic failure of hosts as well as the partitioning of the network are possible. If backups are stored at various sites, it is not possible to guarantee that they are all valid and equally up-to-date. Hence if the primary copy of the data base fails and the backups are inconsistent, one is faced with having to determine which backup (if any) is correct.

A very recent working paper [Johnson and Thomas, 1975] addresses the problem of maintaining duplicate data bases in a network. The authors suppose that there are several sites, each maintaining its own copy of the data, based on update information it receives. Furthermore, each datum is assumed to be a simple (name, value) pair, and the only operations allowed are (1) retrieval of a value, (2) assignment of a new value to a pair, (3) creation of a new pair, and (4) deletion of a pair. To maintain consistency among data base copies, each entry is stored as a triple: (name, value, timestamp), where the timestamp includes both the time of the most recent change and the site originating the change. The time is that when the change originated and is in terms of the originating site's clock. The site information is needed in case clocks at the various sites are not quite synchronized. The authors remark that even if the later of two changes is determined to be the earlier, all sites will make the same decision. Although consistency is maintained, an important update can be lost. Deletions are a problem which is handled by keeping the entry, but flagging it as "deleted". A deleted entry is only removed when its timestamp is sufficiently old.

Concurrent Use. In an environment where data are shared and access is on an interactive basis, it is possible for users to interfere with one another. One obvious source of difficulty is that one user may try to read a location while another is writing it. This problem is usually taken care of by system hardware, which provides an uninterruptible write instruction. For more complex procedures - e.g., rearrangement of data - there is usually a system software provision for flagging a procedure as uninterruptible. In a network situation where there are multiple data copies, the user can take advantage of local non-interrupt facilities to insert the change into an update file

Following this step, a scheme such as that discussed above could be used by the local site to maintain consistency of various data base copies around the network. To the best of our knowledge, there is nothing in the literature which addresses this problem in its entirety.

As soon as a user can be blocked from proceeding with the next step in his process, however, there is the possibility of a deadlock. In the classic deadlock situation, each of two processes ties up a resource needed by the other, so that neither can proceed. A good discussion of the deadlock problem in the context of distributed data bases has recently appeared [Chu and Ohlmacher, 1974]. Deadlock prevention mechanisms usually work by examining the resource requirements of all processes. A process is allowed to proceed only when a deadlock can not occur. If deadlocks are not prevented, they can be detected and resolved by the elimination of one of the processes. This can, unless properly handled, cause considerable user inconvenience.

Deadlock prevention mechanisms usually require that the user specify all files needed for a process. The system can then wait until all files are available before beginning the process. For a network with no central control, Chu and Ohlmacher suggest that the requests for files be passed from node to node in some preassigned order, each node being responsible for granting requests for files at that site.

It may be more efficient to allow a process to proceed unless deadlock is likely to occur (instead of collecting all files needed and holding them for exclusive use of a process). The problem then is to determine whether deadlock is likely. Chu and Ohlmacher suggest that this determination may be made more efficient by grouping all processes having requests for the same file or files into process sets. Requests for shared access and for exclusive access are handled somewhat differently. The result, however, is that the progress of any process is independent of all processes in other sets. The protection mechanism then need only check that no two processes in the same set are run simultaneously.

Chu and Ohlmacher also discuss a simple deadlock detection mechanism. Essentially, it involves keeping a list of processes and pointers to files requested, and testing for the loop configuration which indicates a deadlock. Unfortunately, they see no way to avoid keeping this list at some special network node (perhaps with a backup).

Summary and Assessment

The problem of maintaining data base integrity is in need of considerably more work. At the present time, the principal technique is consistency checking, which usually means devising a set of ad hoc rules, specific to the particular data base, to check the data. This area is in serious need of work to determine whether some more systematic approach is possible.

Serious study of the problem of maintaining backups on a network has only just begun. The work of Johnson and Thomas is very promising but more needs to be done. For example, their limitation of data changes to new assignments of values is quite restrictive. It eliminates such operations as adding items to an inventory or deducting a payment from a billing account. Their method of maintaining consistency by using clock times does, as noted above,

have the disadvantage that if the clocks at different sites are not synchronized an update may be lost. They feel that although the probability of this happening can be made very small, "the distributed nature of the system dictates that this probability can never be zero."

Finally, we have seen that attempts to solve the concurrent use problem have only led to another problem, that of deadlock. Although methods for handling deadlock have been developed, Chu and Ohlmacher's approach to applying these methods in a network environment do not take proper cognizance of the likelihood of site or network failures. This is particularly true of their detection mechanism, which requires a central site to monitor both processes and the files they access.

Introduction

Most work on file allocation has been for single-processor systems. An important aspect of the design of computer operating systems has been the development of algorithms for allocation of the files needed by a program (or programs, in a multiprogramming environment) among the various memory devices available. Analysis has usually relied upon a very simple model, with some sort of mean response time being minimized under a cost constraint. Results generally have indicated that high-usage files should be stored on the fastest device. Although it is an intuitively pleasing rule-of-thumb, this conclusion is questioned in a recent paper that introduces queuing delays into the model [Chen, 1973].

To analyze file allocation in a network environment, an entirely different model must be developed. It is generally no longer valid to assume the existence of a single central processor and optimize from its point of view. (If there is such central control, however, the single processor analyses and results may be applied to the network.) The usual network model assumes that files are distributed around the network and independent processors at various sites are competing for the use of them. Some measure of cost and/or efficiency of the whole network's use of the files must be introduced and used to optimize the system. This area of research is undergoing rapid development at the present time. This report will briefly discuss some of the current research, as well as recent important studies.

Recent and Current Research

Optimal Allocation by Zero-One Programming. The earliest work on the network allocation problem was done by Chu [1969]. Chu states the problem as follows: "Given a number of computers that process common information files, how can we allocate files so that the allocation yields minimum overall operating costs subject to the following constraints: (1) The expected time to access each file is less than a given bound, and (2) the amount of storage needed at each computer does not exceed the available storage capacity." Variables X_{ij} to describe the allocation are introduced; $X_{ij} = 1$ if the j th file is stored in the i th computer and $X_{ij} = 0$ otherwise. In order to apply constraint (1), Chu develops a reasonably comprehensive formula for access time, including queuing delays and the effect of inter-computer traffic congestion. The overall cost expression to be minimized includes costs for storage as well as for transmission. Since the variables to be determined can take on only the values zero or one, the optimal allocation may be found as the solution to a so-called nonlinear zero-one programming problem. (In fact Chu notes that the problem may be reduced to a linear one, which may be solved by straightforward techniques.)

In a later paper [Chu, 1973], he discusses how a reliability constraint can be added to the model. The main idea is to determine in advance (from simple assumptions on failure probabilities) how many redundant copies of a file are required to achieve a desired level of reliability. This number is then

inserted into the model in a simple way, and the basic scheme remains unchanged. The difficulty with using zero-one programming to solve the file allocation problem is that it is so time-consuming as to seem impractical for a large file system. Nevertheless, Chu [1975] tells us that his method has been successfully used at Bell Labs.

Optimal and Suboptimal Allocation by Search Procedures. Believing that zero-one programming is too costly an approach, Casey [1972] developed an efficient search procedure for finding a minimal-cost solution, as well as heuristic methods for finding acceptably good solutions. Casey's model differs in some respects from Chu's. Perhaps the most important difference is that Casey lets the number of copies of a file, as well as its locations, be variables. Notice that as the number of copies of a file increases, the expense of querying the file decreases, but storage and updating costs increase. Thus Casey's approach to optimization strikes a balance between these two opposing trends. A disadvantage, of course, is that the minimization may not yield enough copies for reliability.

Casey has applied his optimization algorithm to real data for the ARPA network and has thus shown the process feasible for networks of moderate size. His experiments indicated that when update traffic equals query traffic, it is most efficient to store all files at a central node. As query traffic increases relative to updates, storage at multiple nodes is indicated. These results are intuitively reasonable. Although one always expects several local minima in a complex, multivariable minimization problem, it is noteworthy that Casey's experiments reveal extremely large numbers of them (over 100 in some cases). It is clear that any optimal allocation procedure must take care to avoid being trapped in such a local minimum.

An interesting extension of this work has been reported [Casey, 1973 (a)]. Casey combines his file allocation algorithm with a well-known procedure for designing tree networks to get a good overall configuration. The method is iterative. An initial network topology is chosen and all files are assumed to be at a central node. Each step then consists of an improvement in network design followed by an optimal file allocation. The method was tried out on an 18-node, 2-file model and results suggested that cost reductions can be achieved in this way. The obvious problem is that use of such a technique in actual network design requires complete a priori information on the files to be stored and their usage.

Recently, both Chu's and Casey's models have been criticized on the basis that they do not allow for dependencies between files and programs [Levin, 1974; Morgan and Levin, 1974]. That is, a program (which is itself a file) may need to make use of one or more data files. The fact that these files must interact with one another is not taken into account in the older models which assume file independence. Morgan and Levin also point out that in a heterogeneous network it may not always be possible to store a particular file at an arbitrary node. Their model takes into account this type of constraint, which also includes the possibility that the allocation may be restricted by security considerations. The algorithm used to solve the optimization problem is a systematic search procedure, along the lines suggested by Casey. Dynamic features were also introduced; that is, costs to change the file allocation were considered and balanced against savings expected from reallocation. Levin [1974] proposes that expected savings may be obtained either from a priori

knowledge of how the queries will probably change with time, or from statistical analysis of the actual queries as they occur.

Some Current Work. Chu [1975] is now working on the so-called catalog problem - i.e., where should the catalog, or file location directory, be put in a network. Under the assumption that there is just one key directory (although with perhaps multiple copies) which all users must query in order to find any file, it is important to the overall efficiency of the system to locate the directory properly. Instead of approaching the problem through a complex mathematical optimization technique (as he did for file allocation), Chu is studying the effects of various trade-offs and trying to develop rules of thumb to indicate where directory copies should be put in various situations.

Kimbleton [1975], working on the general network file allocation problem, claims to have developed an assignment algorithm which is cheap, easy to implement, and intuitively acceptable. His approach to optimal allocation is based on the minimization of "process execution costs." His model therefore includes not only such standard features as file accessing procedures, queuing delays, and cost structures but also information on process "behavior." Kimbleton expects to publish the details of this work in the near future.

Summary and Assessment. The amount of work published to date on the file allocation problem is very small. What little there is suggests certain obvious lines of research which should be pursued. For example: Chu's zero-one programming approach seems too complex for a large data base, but his idea of prescribing a certain amount of redundancy is a good one. On the other hand, Casey's optimization algorithm is efficient, but, by letting the number of file copies be determined by the algorithm, he is likely to arrive at an allocation with insufficient reliability. Perhaps a combination of Casey's efficient algorithm with Chu's reliability constraint could be developed. It should also be possible to include Morgan and Levin's restriction that some files may only be stored at certain nodes, as well as the dynamic features of their work.

Another striking feature of work to date is that costs are always minimized under time constraints. It would seem worthwhile to turn the problem around and minimize average retrieval time under a cost constraint. Trying to get the best possible (in the sense of efficient retrieval) design for your money might be the practical approach in many situations.

References

- Alsberg, P.A.
1975 "Space and Time Savings through Large Data Base Compression and Dynamic Restructuring", to be published in Proc. of the IEEE, Aug. 1975.
- Arora, S.R. and Dent, W.T.
1969 "Randomized Binary Search Technique", CACM 12, pp. 77-80.
- Astrahan, M.M. and Chamberlin, D.D.
1974 "Implementation of a Structured English Query Language", IBM Research Report RJ 1464, October 1974.
- Baum, R.I. and Hsaio, D.K.
1975 "A Semantic Model for Protection Mechanisms in the Data Base System", Proc. Eighth Hawaii Int'l. Conf. on System Sciences, Western Periodicals Company, pp. 175-179.
- Bell, J.R.
1970 "The Quadratic Quotient Method: A Hash Code Eliminating Secondary Clustering", CACM 13, pp. 107-109.
- Berman, G. and Colijn, A.W.
1974 "A Modified List Technique Allowing Binary Search", CACM 21, pp. 227-232.
- Bloom, B.H.
1969 "Some Techniques and Tradeoffs Affecting Large Data Base Retrieval Times", Proceedings ACM 24th National Conf., 1969, pp. 83-95.
- Boyce, R.F. and Chamberlin, D.D.
1973 "Using a Structured English Query Language as a Data Definition Facility", IBM Research Report RJ 1318.
- Cardenas, A.F.
1973 "Evaluation and Selection of File Organization - A Model and System", CACM 16, pp. 540-548.
- Casey, R.G.
1972 "Allocation of Copies of a File in an Information Network", AFIPS Conference Proceedings 40, pp. 617-625.
- 1973 (a) "Design of Tree Networks for Distributed Data", AFIPS Conference Proceedings 42, pp. 251-257.
- (b) "Design of Tree Structures for Efficient Querying" CACM 16, pp. 549-556.
- Chai, D.T.
1974 "Language Considerations for Information Management Systems", ACM Proc. National Conf., 1974, pp. 443-450.

- Chamberlin, D.D. and Boyce, R.F.
1974 "SEQUEL: A Structured English Query Language", IBM Research Report RJ 1394, May 1974.
- Chandra, A.N.
1973 "Some Considerations in the Design of Homogeneous Distributed Data Bases", Proc. Seventh Annual IEEE Comp. Soc. Int'l. Conf., pp. 185-6.
- Chen, P.P.S.
1973 "Optimal File Allocation in Multi-level Storage Systems", AFIPS Conference Proceedings 42, pp. 277-283.
- Chien, R.T. and Mark, E.A.
1974 "A Document Storage Method Based on Polarized Distance", Journal ACM 21, pp. 233-245.
- Chu, W.W.
1969 "File Allocation in a Multiple Computer System", IEEE Trans Computers, C-18 (10), pp. 885-889.

1973 "Optimal File Allocation in a Computer Network", in Computer-Communications Networks, N. Abramson and F. Kuo, eds., Prentice-Hall, Englewood Cliffs, N.J., pp. 82-94.

1975 Private Communication.
- Chu, W.W. and Ohlmacher, G.
1974 "Avoiding Deadlock in Distributed Data Bases", Proc. ACM Nat'l. Symp. 1, Nov. 1974, pp. 156-160.
- CODASYL Data Base Task Group
1971 "April 71 Report"
- Codd, E.F.
1970 "A Relational Model for Large Shared Data Banks", CACM 13, pp. 377-387.

1971 "Normalized Data Base Structure: A Brief Tutorial", Proc. 1971 ACM - SIGFIDET Workshop, pp. 1-18.

1972 "Relational Completeness of Data Base Sublanguages" in Data Base Systems, R. Rustin, ed., Prentice-Hall, pp. 65-98.

1974 "Recent Investigations in Relational Data Base Systems" IBM Research Report RJ 1385, April, 1974.
- Coffman, E.G., Jr. and Eve, J.
1970 "File Structures Using Hashing Functions", CACM 13, pp. 427-432, 436.
- Crouch, D.B.
1973 "A Process for Reducing Cluster Representations and Retrieval Costs" ACM Proceedings Annual Conference, 1973, pp. 224-227.

- Dearnley, P.A.
1974 (a) "A Model of a Self-organizing Data Management System", Computer J. 17, pp. 13-16.

(b) "The Operation of a Model Self-organizing Data Management System", Computer J. 17, pp. 205-210.
- Dennis J.B.
1970 "On the Exchange of Information", ACM SIGFIDET Workshop, pp. 41-67.
- Dostert, B.H. and Thompson, F.B.
1971 "How Features Resolve Syntactic Ambiguity", Proc. Symp. on Inform. Stor. and Retrieval, pp. 19-32.
- Earley, J.
1971 "Toward an Understanding of Data Structures", CACM 14, pp. 617-627.
- Finkel, R.A. and Bentley, J.L.
1974 "Quad Trees: A Data Structure for Retrieval on Composite Keys", Acta Informatica 4, pp. 1-9.
- Florentin, J.J.
1974 "Consistency Auditing of Databases", Computer J. 17, pp. 52-58.
- Flores, I.
1970 Data Structure and Management, Prentice-Hall.
- Frank, R.L. and Yamaguchi, K.
1974 "A Model for a Generalized Data Access Method", AFIPS Conference Proceedings 43, pp. 45-52.
- Friedman, T.D.
1970 "The Authorization Problem in Stored Files", IBM Systems Journal 9 pp. 258-280.
- Ghosh, S.P.
1972 "File Organization: The Consecutive Retrieval Property", CACM 15, pp. 802-808.
- Gorenstein, S. and Galati, G.
1974 "Data Base Reorganization for a Storage Hierarchy", IBM Research Report RC 5063, October 1974.
- Gotlieb, C.C. and Tompa, F.W.
1974 "Choosing a Storage Schema", Acta Informatica 3 (1974), pp. 297-319.
- Hoffman, L.J.
1971 "The Formulary Model for Flexible Privacy and Access Controls", AFIPS Conference Proceedings 39, pp. 587-601.
- Hopgood, F.R.A. and Davenport, J.
1972 "The Quadratic Hash Method When the Table Size is a Power of 2" Computer J. 15, pp. 314-315.

- Hsiao, D.K.
1971 "A Generalized Record Organization", IEEE Transactions on Computers, C-20, pp. 1490-1495.
- Hsiao, D. and Harary, F.
1970 "A Formal System for Information Retrieval from Files", CACM 13, pp. 67-73.
- Huang, J.C.
1973 "A Note on Information Storage and Retrieval", CACM 16, pp. 406-410.
- Huffman, D.A.
1952 "A Method for the Construction of Minimum Redundancy Codes", Proc. I.R.E. 40, 1028-1101.
- Jardine, N. and van Rijsbergen, C.J.
1971 "The Use of Hierarchic Clustering in Information Retrieval", Inform. Stor. Retr. 7, pp. 217-240.
- Johnson, P.R. and Thomas, R.H.
1975 "The Maintenance of Duplicate Databases", ARPA Network Working Group RFC #677 (NIC #31507).
- Kimbleton, S.
1975 Private Communication
- Lefkovitz, D.
1969 File Structures for On-line Systems. Spartan Books, New York.
- Levin, K.D.
1974 "Organizing Distributed Data Bases in Computer Networks", Ph.D. Dissertation, University of Pennsylvania, 1974.
- Linde, R.R., Gates, R. and Peng, T.
1973 "Associative Processor Applications to Real-Time Data Management" AFIPS Conference Proceedings 42, pp. 187-195.
- Lum, V.Y.
1970 "Multi-attribute Retrieval with Combined Indexes", CACM 13, pp. 660-665.

1973 "General Performance Analysis of Key-to-Address Transformation Methods Using an Abstract File Concept", CACM 16, pp. 603-612.
- Lum, V.Y. and Yuen, P.S.T.
1972 "Additional Results on Key-to-Address Transform Techniques: A Fundamental Performance Study on Large Existing Formatted Files" CACM 15, pp. 996-7.
- Lum, V.Y., Yuen, P.S.T. and Dodd, M.
1971 "Key-to-Address Transform Techniques: A Fundamental Performance Study on Large Existing Formatted Files", CACM 14, pp. 228-239.

- Maurer, W.D.
 1968 "An Improved Hash Code for Scatter Storage", CACM 11, pp. 35-38.
 1969 "File Compression Using Huffman Coding" in Computing Methods in Optimization Problems - 2 (L.A. Zadeh, L.W. Neustadt and A.V. Balakrishnan, eds.), Academic Press, N.Y.
- McCarthy, J.P.
 1974 "Automatic File Compression" in International Computing Symposium 1973 (A. Gunther, B. Levrat and H. Lipps, eds.), American Elsevier, N.Y.
- Mommens, J.H. and Raviv, J.
 1974 "Coding for Data Compaction", IBM Research Report RC 5150, Nov. 1974.
- Morgan, H.L. and Levin, K.D.
 1974 "Optimal Program and Data Locations in Computer Networks", Report 74-10-01, Dept. of Decision Sciences, The Wharton School, University of Pennsylvania.
- Morris, R.
 1968 "Scatter Storage Techniques", CACM 11, pp. 38-43.
- Moulder, R.
 1973 "An Implementation of a Data Management System on an Associative Processor", AFIPS Conference Proceedings 42, pp. 171-176.
- Mulford, J.B. and Ridall, R.K.
 1971 "Data Compression Techniques for Economic Processing of Large Commercial Files", Proceedings of the Symposium on Information Storage and Retrieval, ACM, 1971, pp. 207-215.
- Mullin, J.K.
 1972 "The Specification of Data Structures, Access Methods, and Efficiency", Proceedings Sixth Annual Princeton Conference on Information Sciences and Systems, pp. 79-84.
- Nishihara, S. and Hiroshi, H.
 1974 "A Full Table Quadratic Search Method Eliminating Secondary Clustering", Int. J. Comp. Inform. Sci. 3, pp. 123-128.
- Page, E.S. and Wilson, L.B.
 1973 Information Representation and Manipulation in a Computer, Cambridge Univ. Press.
- Patt, Y.N.
 1969 "Variable Length Tree Structures Having Minimum Average Search Time", CACM 12, pp. 72-76.
- Ramamoorthy, C.V. and Chin, Y.
 1971 "An Efficient Organization of Large Frequency-Dependent Files for Binary Searching", IEEE Trans. Comp. C-20, pp. 1178-1187.

Reardon, B.C.

1974 "An Adaptive Information Retrieval System Using Partial File Inversion", Inform. Stor. Retr. 10, pp. 49-56.

Rettenmayer, J.W.

1972 "File Ordering and Retrieval Cost", Inform. Stor. Retr. 8, pp. 79-93.

Rothnie, J.B., Jr. and Lozano, T.

1974 "Attribute Based File Organization in a Paged Memory Environment", CACM 17, pp. 63-69.

Rotwitt, T., Jr. and deMaine, P.A.D.

1971 "Storage Optimization of Tree Structured Files Representing Descriptor Sets", Proceedings of 1971 ACM Sigfidet Workshop (E.F. Codd and A.L. Dean, eds.), pp. 207-217.

Ruth, S.S. and Kreutzer, P.J.

1972 "Data Compression for Large Business Files", Datamation, Sept., 1972, pp. 62-66.

Salton, G.

1968 Automatic Information Organization and Retrieval, McGraw-Hill.

Schuster, S.

1974 Private Communication.

Schwartz, J.T.

1972 "Abstract and Concrete Problems in the Theory of Files" in Data Base Systems, R. Rustin, ed., Prentice-Hall, pp. 1-22.

Senko, M.E.

1972 "Details of a Scientific Approach to Information Systems" in Data Base Systems, R. Rustin, ed., Prentice-Hall, pp. 143-174.

Senko, M.E., Altman, E.B., Astrahan, M.M. and Fehder, P.L.

1973 "Data Structures and Accessing in Data Base Systems", IBM Systems Journal 12, pp. 30-93.

Senko, M.E., Lum, V.Y. and Owens, P.J.

1969 "A File Organization Evaluation Model (FOREM)", Information Processing 68, North-Holland, Amsterdam, pp. 514-519.

Sibley, E.H. and Taylor, R.W.

1973 "A Data Definition and Mapping Language", CACM 16, pp. 750-759.

Skinner, C.W.

1974 "A Heuristic Approach to Inductive Inference in Fact Retrieval Systems", CACM 17, pp. 707-712.

Stanfel, L.E.

1973 "Optimal Trees for a Class of Information Retrieval Problems", Inform. Stor. Retr. 9, pp. 43-59.

- Stocker, P.M. and Dearnley, P.A.
1973 "Self-organizing Data Management Systems", Computer J. 16, pp. 100-105.
- Stonebraker, M.
1974 "The Choice of Partial Inversions and Combined Indices", Int. J. Comp. Inform. Sci. 3, pp. 167-188.
- Summers, R.C., Fernandez, E.B. and Coleman, C.D.
1975 "Shared Data Access Control with Programming Language Support", Proc. Eighth Hawaii Int'l. Conf. on System Sciences, Western Periodicals Company, pp. 187-190.
- Sussenguth, E.H., Jr.
1963 "Use of Tree Structures for Processing Files", CACM 6, pp. 272-279.
- Thompson, F.B., Lockemann, P.C., Dostert, B. and Deverill, R.S.
1969 "REL: A Rapidly Extensible Language System", ACM Proc. National Conf., 1969, pp. 399-417.
- Traiger, I.
1975 Private Communication.
- Van Rijsbergen, C.J.
1971 "An Algorithm for Information Structuring and Retrieval", Computer J. 14, pp. 407-411.

1974 "Further Experiments with Hierarchic Clustering in Document Retrieval", Inform. Stor. Retr. 10, pp. 1-14.
- Vold, H. and Sjogren, B.H.
1973 "Optimal Backup of Data Bases: A Statistical Investigation", BIT 13, pp. 233-241.
- Winkler, A.J.
1971 "File Structure Determination", Proceedings ACM Symposium Inform. Stor. and Retr., J. Minker and S. Rosenfeld, eds., pp. 133-146.
- Wong, E. and Chiang, T.C.
1971 "Canonical Structure in Attribute Based File Organization", CACM 14, pp. 593-597.
- Yue, P.C. and Wong, C.K.
1974 "Storage Cost Considerations in Secondary Index Selection", IBM Research Report RC 5070, October 1974.

Network and Systems Environment

Overview

The successful construction of data management systems on a computer network is heavily dependent on the existence of a resource sharing network environment. In this section we survey the essential aspects of the necessary technology. The treatment is divided into 5 sections:

- 1) Communications and Networks
- 2) Resource Allocation and Control
- 3) Measurement and Analysis
- 4) Network Access Systems and Front-Ends
- 5) Security

The state-of-the-art in computer communications and networks has greatly advanced in recent years, largely due to the amount of effort being expended on both commercial and government-sponsored networks. The design of computer communication networks has progressed faster than the technology of distributed system design. This is because emphasis is usually placed on making the communication network work as soon as possible. Now that the feasibility of computer networks has been demonstrated, work is progressing on other aspects of networking, such as security and resource sharing.

The decisions governing the resource sharing of a particular task in a computer network depend heavily on the ability to measure the performance of the network and its component systems. These measures provide the necessary information for a task to be performed at the lowest price or in the shortest amount of time.

The advent of the computer network has seen some computer centers come to rely entirely on a network for their computing facilities. This has led to research on network access systems which provide inexpensive and flexible network access for terminals and file systems. These same techniques can be used to front-end large machines that are unsuited or too heavily utilized to support network software.

Introduction

The need for digital communications has grown enormously in recent years. This reflects, in part, a growing realization of the breadth of the field of computer applications. Many of these applications require immediate access to geographically remote resources, such as data, computers, and people. The need is hardly a new one, but its magnitude has increased quite unexpectedly. This is probably due in large part to the plummeting price and subsequent proliferation of computers in the last 25 years. John Von Neumann once expressed the opinion that the computing needs of the U.S. could be filled by two large computers, one on each coast. The thrust of current technology is to place one in every home, office, automobile, and pocket. These computers must communicate to realize their full potential. This section describes some aspects of the current state of the art in computer communications, with emphasis on general-purpose computer networks and packet-switching techniques.

The first section, Communication Media, defines some terminology used in describing communication facilities, then describes some of the available and proposed common-carrier and value-added carrier facilities in the United States. A related topic, broadcast communication media, is described in the Multiple-Access Broadcast, Radio, and Satellite Communication Section.

The next section, Communication Network Design Issues, defines some of the basic criteria used to describe and evaluate communication networks, then describes the current technology of network construction. The topic areas are Delay, Bandwidth, Reliability, Cost, Packet Switching, Forwarding Schemes, Packet Size, Store-and-Forward Switching Node Design, Topology, Routing, Flow Control, and Network Analysis and Modeling.

The third section is Multiple-Access Broadcast, Radio, and Satellite Communications. This section describes a technology with great potential for reduction of cost of personal and computer communication. The media and techniques are described, and some current work is briefly examined.

The final section is a Summary and Assessment of the current state of the art in computer communications. Some areas in which there is a need for further research and study are indicated.

Communication Media

Introduction

The mechanisms used to communicate digital data from point to point will be referred to as communication media. A variety of media are in use today. For short distances, radio, light, and cables can be used conveniently. For longer distances, the expense of obtaining right-of-way for cables, the power and licensing problems involved for high-power radio, and the line-of-sight properties of light and high-frequency radio virtually preclude the establishment of private communications facilities by any but the largest concerns. For these reasons, most computer networks will be constructed using

facilities leased from common-carrier communication companies. This section briefly looks at the current state of technology in the US communications industry.

Communication facilities can be analog or digital in nature. Analog communication is well suited to voice and some facsimile use, as it allows transmission of continuously variable signals. High error rates do not interfere greatly with human understanding of voice and pictures, hence most analog systems have not been designed to provide distortionless, error-free communication as required by computers. Furthermore, since analog data cannot easily be reconstructed and "cleaned up" at repeaters, noise added by quantum effects, circuit noise, and other spurious signals accumulate with every repeating. Digital communication involves the transmission of only two signal levels, usually referred to as zero and one. Digital systems are designed to be used by computers, hence their error rates tend to be lower by design. Digital signals can also be "cleaned" of any spurious noise, provided the noise is distinguishable from zero and one, at repeaters.

Digital or analog communication networks can be either circuit-switched (line-switched) or message-switched. Circuit-switched networks, such as the telephone system, establish a fixed path through the network capable of passing from zero to the full bandwidth desired. This set-up usually involves operations such as selecting appropriate lines and subchannels on multiplexers to route the data from the source to the desired destination. Circuit switching involves a variable connection set-up time (very long if manually done, otherwise usually on the order of seconds) and a fixed (for each set-up) transmission delay (usually essentially speed-of-light limited). In most applications, circuit-switched lines carry useful data a relatively low percentage of the time, but use the same resources regardless of utilization. Message-switched networks break the data sent into pieces (messages) and send these messages into the network to follow whatever route seems best at that instant. If messages are explicitly stored at any point in the path, the network is referred to as a store-and-forward network (the usual case). Message switched networks use the bursty nature of the data they carry to increase average line utilization. However, because of routing, queueing (which occurs in the store-and-forward case), and transmission delays, the exact delay between transmission and receipt of a message normally varies for different messages. A comparison of circuit and packet switched (store-and-forward with fixed message (packet) lengths) network traffic capacity can be found in [Itoh and Kato, 1973]. Message-switched networks typically achieve many orders of magnitude higher reliability by the use of feedback-error-correction techniques and adaptive routing (to circumvent failing components) [Roberts and Wessler, 1970].

Communications networks which utilize other networks in providing their services are referred to as value-added carriers. The "value-added" refers to the fact that while existing carrier facilities are used, more services of some sort are provided, increasing the value (and cost) of the existing facilities to an end user. An example is a digital store-and-forward network built upon analog circuit-switched communication facilities.

Common Carrier Communication Utilities

In order to encourage competition with communications giants such as AT&T, the Federal Communications Commission (FCC) licensed several companies to sell communication services in interstate commerce. The salient features of the most prominent networks are summarized.

Telephone Network. The use of the world-wide telephone network as a digital data transmission network is fairly well understood. The majority of current literature in this area concerns either choosing the most appropriate services to fill the needs for a given application or minimizing the cost of such services by data compression, multiplexing, and higher level technologies such as packet switching. Since the telephone network is fundamentally analog in nature, various encoding schemes are used to transmit digital information. Problems caused by the voice orientation of the system include reserved audio frequencies used by the network switching equipment (which must be avoided), phase shift, echoing (particularly over very long distances), and highly correlated errors (burst errors) which can render conventional error correction techniques ineffective. Recommendations are made by the Comité 'Consultatif Internationale' de Telegraphie et Telephone (CCITT), part of the International Telecommunications Unions (an organ of the United Nations), concerning use of the telephone system for digital transmission. While these recommendations are not standards, they are generally followed. Authorities agree that digital networks of the flexibility of the telephone system are badly needed now and will be needed even more in the future to meet the data communications needs of the industry. [Davies and Barber, 1973], [Lucky, 1973], [Hansler, 1974], [Chou and Kershenbaum, 1973], [Cohen, 1974].

Datran. Datran was founded in 1968 and started construction of its entirely new microwave network in 1973. The network will connect both coasts by 1975, but will not be fully articulated, with all spurs and cities covered by that time. Some facilities are being shared with SPCC (see below). The network primarily connects large population centers.

Datran is an all-digital, circuit-switched network. Datran will offer switched, medium-speed (2.4 - 9.6 Kbaud) service in early 1975. High-speed (56. - 2,688 Kbaud) private channel service is also available. Reliability is expected to be high, with an error rate on the order of 1 error in 10^7 bits.

Connection in the low-speed switched service is expected to take .5 seconds from completion of dialing. Switched service is currently being considered for the high-speed service.

SPCC - Southern Pacific Communications Company. SPCC is an outgrowth of the Southern Pacific Company's private communications system. SPCC is an analog system providing only private-channel, directly-connected service. Digital service is provided by conventional techniques for utilizing analog circuits. Leased satellite communications were to be incorporated into the network in mid-1974 to complement the terrestrial facilities. SPCC intends to provide service to population centers in the east, west, and south central United States.

Primary digital offerings are for narrow band (voice grade line) to medium speed (9.6 Kbaud) service. Higher bandwidth can be obtained by patching together voice grade channels. The design objective for error rate is 1 bit in 10^6 . Past availability of the Southern Pacific Company's private network has been 99.96%. The SPCC objective is 99.98%, using newer equipment and designs.

WTCI - Western Telecommunications Incorporated. WTCI operates primarily in the West, with a leased satellite link from southern California to

New York and Chicago. WTCI specializes in analog services such as video, facsimile, etc., but will provide modems, etc. for digital use. WTCI provides analog service from voice to 240 KHz, and digital service from 75 baud to 50 Kbaud. Circuits are private leased line and provide no switching capability. WTCI has informally announced plans to offer a message-switched service, but has not released details yet.

Value-Added Carriers

PCI - Packet Communications Incorporated. PCI was the first organization to apply to the FCC for permission to offer a packet switched communication service. The PCI network is heavily based on ARPA network technology, which is predictable since PCI was founded primarily by people from Bolt, Beranek, and Newman (BBN), prime ARPA network contractors. PCI estimates that their system will be available by late 1975. PCI's primary market is low-bandwidth, low-delay, interactive terminal-like applications. PCI is technically supported by Network Analysis Corporation, prime analysis contractor for the ARPA network and a co-founder of PCI.

The PCI network closely resembles the ARPA network in structure. Message switching processors called PSP's (Packet Switching Processors) connect 50 Kbaud leased lines as ARPA network IMP's [Heart et al., 1970] do. Direct terminal access to the network is provided by a TAP (Terminal Access Processor), the counterpart of the ARPA network TIP (Terminal Interface Processor) [Mimmo et al., 1973]. The ARPA TIP is an expanded IMP which can handle terminals and packet switching. The TAP requires connection to a PSP. This is more like the ARPA network mini-host IMP architecture [ANTS, 1973]. Customers can use the facilities of the network with any of three levels of software interface, varying from an ARPA-network-like NCP (Network Control Program) to a very minimal level.

The network is expected to be 99.5% available, except for scheduled maintenance, with a reliability of 1 undetected bit error per year. Round trip delay between two interfaces is expected to be .5 second or less. These figures closely resemble the original ARPA network specifications.

Telenet Communications Corporation. Telenet is a subsidiary of Bolt, Beranek, and Newman (BBN), prime contractors for the ARPA network. Its president is Larry G. Roberts, past director of the Information Processing Techniques Office of ARPA, under whose supervision the ARPA network was built. The Telenet network is heavily based on ARPA network technology, and continues to benefit from such current work as the Satellite IMP (SIMP) being developed at BBN.

The Telenet network will initially connect seven large cities (Boston, Chicago, Dallas, Los Angeles, New York, San Francisco, and Washington) with 50 Kbaud common or special carrier lines. A Central Office (C.O.) will be located at each major city. The C.O. will contain IMP's and TIP's for that area, along with more customized customer service facilities. Users will generally connect to a nearby C.O. with leased lines or dial to a TIP via switched telephone service. If usage warrants it, a TIP may be leased to a customer and located at his site.

Telenet will attempt to provide interfaces to existing systems that require little or no modification to vendor software. Customers may choose to modify their systems if they desire. The network uses ARPA network concepts such as the Network Virtual Terminal, modified to better suit the commercial market situation.

Telenet also feels that there is a broad market for wideband digital services. They are proceeding with development of satellite technology to provide the capacity needed in the future. Higher bandwidth interconnection among IMP's will be used as demand increases. Satellite channels will be random-access broadcast channels at 1,544 Kbaud. Telenet claims an error rate of under 10^{-12} , or one bit error in 10^{12} bits, which is 5 orders of magnitude better than circuit-switched specialty carriers such as Datran. Messages up to 100 characters long will traverse the network within .3 second. Longer messages, up to 1000 characters, will require .8 second or less.

Communication Network Design Issues

Delay

Delay is the time between the sending of a message and its receipt. Delay is a function of communication line length and speed, packet length, number of hops (i.e. network nodes) in the path taken by the message, and processing time and queuing delays in switching nodes. For highly interactive use of a network, the delay should be small. Unfortunately, small delay requires small packets, high bandwidth lines, a small number of hops, or some combination of these. Small packets increase overhead per data bit, thus reducing the effective capacity of the line. Very fast lines and fewer hops cost money. An analysis of these tradeoffs can be found in [Metcalf, 1973]. The ARPA network, which was designed with delay as a primary consideration, averages about 5 hops, uses primarily 50 Kbaud lines, and uses 1000-bit packets. The coast-to-coast delay for one packet is under .5 seconds, and average delay is much less. See also: [Kleinrock, 1973], [Frank, 1972], [Frank and Chou, 1974].

Bandwidth

Bandwidth is the average number of bits per second transferred over a data link in some interval. A very important bandwidth measure of a network is peak bandwidth, the highest bandwidth the network can sustain for a short interval. The peak bandwidth is determined primarily by line speed and line overhead. In actual measurements taken on the ARPA network, Kleinrock, Naylor, and Opderbeck [1974] found that while the maximum possible line utilization was about 80%, current short message traffic would limit it to 17%. Clearly, maximum bandwidth can only be achieved by using a network in an "optimal" fashion, which may not be possible in general [Cerf, 1974].

It is, at least in theory, possible for the packets traveling between two nodes to travel "in parallel", different packets using different paths through the network. Thus, two nodes can communicate at aggregate speeds higher than they could on any single connecting line. The highest possible bandwidth is achieved by choosing the highest speed paths through the network and parceling the data appropriately among them. See the Routing section for further discussion.

Reliability

The reliability of a network has two components: data reliability, the probability of undetected data errors; and availability, the percentage of real time the network is usable. The availability of the ARPANET averaged about 98% for 1972 - Apr. 1974 while commercial networks are planning for over 99.5% availability [Cohen, 1974]. The ARPANET data reliability for received packets was targeted at 1 undetected bit error per year, but packets are sometimes completely lost and host-to-IMP interfaces have no error detection mechanism. Hence observed data reliability is considerably under the target value. ARPANET protocols assume the network is reliable to a very large degree, and hence often cannot cope with network generated errors. A modification to ARPANET protocols to correct this has been suggested [Kanodia, 1974], and an alternate protocol has also been suggested [Cerf and Kahn, 1974].

Availability is a relative measure. If one node is isolated from the rest of the network, the availability of the network with respect to that node is temporarily 0%, while the rest of the network may be otherwise unaffected. This is a special case of network partitioning, the separation of a network into two or more non-communicating networks. The probability of disconnection of parts of a network is a factor of the reliability of nodes and communication lines, and of the "topology", or interconnection, of the network. The probability of isolation of a particular node is always greater than $P_n + P_1^N$, where P_n is the probability of a node failure, P_1 is the probability of a line failure, and N is the number of lines to the node. The node reliability problem is being considered by Bolt, Beranek, and Newman [Heart et al., 1973]. The line reliability is largely in the hands of the common carrier chosen. The topology of networks, as it affects availability, has been studied in some detail by the Network Analysis Corporation and others (see Topology section). In general, networks should be rather highly interconnected, though only careful analysis can determine the best tradeoff between cost and interconnection. See the section on Network Analysis and Modeling for further discussion.

Cost

The cost of a communications network is intimately tied to every other parameter. The cost of each component should be kept at a minimum, given performance constraints. In the case of switching nodes, the peak load through a node is estimable, so minimal-cost nodes could theoretically be installed [Heart et al., 1973]. Similarly, communication lines can be configured in such a way that cost is minimized without sacrificing reliability [Frank and Chou, 1974].

Packet Switching

The basic principle of store-and-forward networks is that units of data, instead of passing directly from sender to receiver, go through a series of intermediate destinations known as store-and-forward switches or nodes. Each of these nodes verifies that it has received the information correctly (using appropriate error detection techniques), then sends it one step closer to its destination. If the unit of data is an essentially arbitrary length message, as in the military AUTODIN network, the node must have sufficient

storage to store the entire message. Long messages can also tie up communication channels for long periods of time, thus increasing delay. For this reason, the idea of breaking messages up into small, uniform-length pieces, known as packets, arose. Most contemporary networks used for low-delay communication use packet switching technology, and for this reason, we will concentrate our efforts in this area. Where unambiguous, we often use the words message and packet interchangeably.

Forwarding Schemes

Using a hop-by-hop acknowledgement scheme, each node of a store-and-forward network refuses to take responsibility (via positive acknowledgement) for any message until it has CORRECTLY received it. Each node is required to retain its copy of the data until it has forwarded the data and received an acknowledgement. If a message is not acknowledged within some time interval (a "timeout" period), the station retransmits it. In an end-to-end acknowledgement scheme, the message is sent from node to node, as above. However, if a node finds that a transmission error has corrupted the data, it discards it. In an end-to-end acknowledgement scheme, individual nodes do not have responsibility to determine that messages have been successfully forwarded. Thus, a piece of data entering the network may be totally lost. It is the responsibility of the destination node to acknowledge correctly received data. It is the sending node's responsibility to retransmit data if no acknowledgement is received in a "reasonable" amount of time. If the number of hops or the probability of transmission errors is large, hop-by-hop acknowledgement yields slightly lower transfer times [Metcalfe, 1973]. This result would be even stronger if the mechanism used to detect failure in the end-to-end case, e.g., timeout or negative acknowledgement, were included in the end-to-end times. There is presently some controversy over the relative merits of these two schemes. The end-to-end advocates argue that computers communicating via store-and-forward networks should perform higher level end-to-end checking anyway. This precludes the possibility that lost or duplicated data will go undiscovered. Complete loss of data could occur, for example, if a hop-by-hop node became isolated or broken while retaining sole copies of messages. Duplication can also occur in several ways. End-to-end checking reduces the need for a hop-by-hop acknowledgement. The complexity of the node is also reduced. However, the intercommunicating computers must now do more work. Both sides have strong arguments, and the resolution of this problem may be political rather than technical. It is significant that the ARPA network architects (Bolt, Beranek and Newman) have recently added an "uncontrolled packet" feature to the ARPA network primarily for use in speech communication experiments, but presumably also in deference to the controversy. The uncontrolled packet is considered expendable, and is discarded if problems arise (e.g., if it is corrupted in transit).

The topic of routing messages between nodes, i.e., deciding which neighboring node to forward a message to next, is covered in a separate section on Routing.

Packet Size

Packet size refers to the length of the series of bits that is treated as a unit in a packet switched network. It is generally the buffer size in the switching node. Packet size must be determined by considering the desired delay (reducing packet size reduces delay caused by transmission

time), error probability (larger packets increase the probability and cost of retransmission), desired peak bandwidth (longer packets mean less overhead), and switching node memory cost. Considering ARPANET error rates, line speed, and fixed overhead, Metcalfe [1973] arrives at an optimum packet size of 3700 bits. This figure optimizes bandwidth but ignores delay. On the other hand, the Canadian BNR network, optimized for low-delay terminal use, uses a packet length of about 350 bits [Martel et al., 1974]. Davies and Barber [1973] recommend a packet length of 2000 bits for a general-purpose network. This is also the figure used by the Canadian DATAPAC network [DATAPAC, 1974]. The ARPA network uses a packet length of 1000 bits, which is a compromise between delay and bandwidth, but introduces the concept of a "message". A message, in this sense, is composed of 8 or less packets. The BNR network also uses this concept, but a message is four or less packets. A computer using such a network can send and receive these messages independently of the network packet size. The communications node has the responsibility to break transmitted messages into packets and to reassemble packets into messages at the receiving end (message reassembly). A well-known deadlock known as "reassembly lockup" can occur if buffer space is scarce in the receiving node [McQuillan et al., 1972]. This problem can be circumvented, but many people still argue that message reassembly should be the task of the intercommunicating computers, not the communications subnetwork. Both sides have good arguments, and further research is needed. For details of these arguments, see [Cerf, 1974] and [Crowther et al., 1974].

Store-and-Forward Switching Node Design

The store-and-forward node has the responsibility for formatting, routing (forwarding), and receiving packets. Since a switching node can send data through several (generally at least two) communication lines, and similarly can receive data simultaneously from several lines, the throughput through the node could reach four times the line speed (assuming full-duplex lines and two neighboring nodes) without any data being transferred into or out of a computer connected to the node. If all traffic leaving the node was generated by the computer connected to it, and all traffic entering the node was destined for the computer, the entire traffic of the node could enter and leave the computer (assuming a full-duplex interface operating at twice line speed), for a total of eight times the line speed. This places a heavy burden on the switching node. The node also must perform chores such as maintaining tables of routing information, checking communications lines (and nodes), performing measurement and accounting, and checking its own operation. All these functions are complex and operate under severe time and space constraints. For ease of maintenance, all nodes should be identical, or virtually so. These constraints indicate that node design and maintenance could most easily be carried out if one group had responsibility for the nodes, and that allowing (or requiring) each node to be programmed by its owners or users would be unreliable and inefficient. [Davies and Barber, 1973], [Heart et al., 1973], [Heart et al., 1970], [Pouzin, 1973 (a) and (b)], [Carter, 1973], [Crowther et al., 1973]

Topology of Communication Networks

The topology of a network is the placement and interconnection of its nodes. This section will summarize the major classes of topologies and their salient features.

Star networks are the least connected of all topologies. A simple star network has a single central node and an arbitrary number of nodes (points) connected solely to the central node. The central node can be the sole intelligence in the network (as in a network of terminals), a "distinguished" member of a set of computers, or a simple message switch, serving only to connect the other nodes together. The major problems with a star network are reliability and the bandwidth requirements of the central node. Since every node is connected to the central site by a single line (i.e., it is 1-connected), the failure of that line isolates the node. Since all nodes (except the central one) are 1-connected, the probability of the loss of communication between two nodes is the sum of the probabilities of failure in the two communicating nodes, the central node, and the two connecting lines. Since the central node must take some part in the communication between two nodes, it must be able to handle the aggregate bandwidth of all communications, or control hardware that does the work. In either event, large networks require large central nodes with correspondingly high probability of failure.

The major advantage of star networks is centralized control. The central node can handle resource allocation, serve as a mediator for establishing communications, keep statistics and accounting information on data transferred, serve as a repository for the location mechanism for resources and processes, and so on. The advantages of centralized control can be obtained in any topology. Networks with a central controller but higher connectivity than a star network are called centralized networks [Frank and Chou, 1972]. A simple case is a hierarchy of concentrator stations feeding a central computer. A more complex example might be a "logical star" topology, set up within a more general environment such as the ARPA network. In this scheme, some computer is designated the controller, and all other computers communicate solely with it. Such a network would be able to benefit from the added reliability of a multiply-connected communication network and the simplicity of central control.

When star networks are interconnected, some new problems can arise because there is no longer a unique central node. These problems are addressed in the discussion of arbitrarily interconnected networks.

Ring networks or loop networks consist of a number of nodes connected to form a closed loop. Each node has one input line and one output line. More complex arrangements have been described, but this model is sufficient for discussion. In the usual model, empty packets are circulated in the loop in timed intervals called slots. Each slot is one packet in length (much like the "Lazy Suzan" serving tray). The ring may also operate in a buffered mode ([Hayes and Sherman, 1971], [Kaye, 1972], [Farmer and Newhall, 1969]) but the distinction is unimportant here. When a node wants to send a packet, it waits until it sees an empty slot at its input line and replaces the empty packet with the one it wishes to send. The packet includes the source and destination node addresses. When a node notices a non-empty packet entering on its input line, it checks the destination address. If the address is its own, the node copies the packet into its own memory and marks the packet as received. Nodes also watch the source address of non-empty packets to see if the packet was sent by themselves. If it was, the packet is checked to be sure it was received; if so, the slot is marked empty, removing the packet from the loop. If not, it must be re-sent.

Many details have been omitted from this scenario. For instance, packets must be checked for correctness in some way by the receiver. If the packet does not check out, the receiver may simply refuse to mark it received. The sender, upon finding an unreceived packet, would retransmit a correct copy. Packets may also be damaged in such a way that neither sender nor receiver recognizes them. Such packets could occupy a slot forever if no mechanism were provided to remove them.

Ring networks have many shortcomings. The reliability is theoretically low, since any break ruptures the ring. Circuitry to automatically bypass failing nodes reduces this to the probability of line failure, provided the bypass is failproof. Some ring networks can also take advantage of dial-up common carrier communications, which would reduce the impact of simple line failures. A double-ring architecture has been proposed to raise reliability [Farber, 1972], but this solution raises problems of its own (such as cost). Since the probability of line errors is also cumulative, a network of 20 nodes will experience an order of magnitude more packet errors between two nodes than if communication were done over a direct line. Since feedback error correction requires an entire trip around the ring, delay could sometimes be very high. A mechanism to prevent infinitely circulating packets, as described above, must be provided. Since all traffic must pass over every communication link, the potential bandwidth requirements of the links are rather high. Interconnecting several smaller rings can help combat this problem (see [Pierce, 1971] or [Hayes and Sherman, 1971]).

Despite these shortcomings, ring networks provide advantages that can make them favorable for some applications. By providing a special "broadcast" address or bit, packets can be sent to all nodes at no extra cost. However, since packets may be transmogrified in transit, it is difficult to guarantee that all nodes got such a message unless the number of nodes is known. Because no node needs special knowledge, other than its own address, the exact size of the ring is immaterial and can be easily changed. If, as is done in DCS [Farber and Larsen, 1972], an associative memory is placed in each node, destination addresses can be associated with movable objects, such as processes, instead of nodes. This technique allows one to bypass many of the sticky problems in distributed architectures and to proceed to higher level, more interesting problems (e.g., process migration, distributed file systems). In addition, ring hardware interfaces can be constructed very cheaply. This is a major factor when the network will not be geographically dispersed and an inexpensive, reliable communications medium, such as co-axial cable, is available [Loomis, 1973], [Farber, 1972], [Farber and Larson, 1972], [Hassing et al., 1973], [Spragins, 1972], [Hayes and Sherman, 1971], [Kaye, 1972], [Yuen et al., 1972].

Completely connected networks are networks in which every node is connected to every other node. Broadcast networks are a special case, and are discussed in their own section. This interconnection scheme is very fast, potentially very reliable, and very expensive. While problems such as line outages can cause loss of a direct connection between two nodes, alternate routes through other nodes are very likely to exist. Because of the expense of complete interconnection, it is most practical for small networks and networks in which the cost of a communications link can be made very low, such as geographically centralized networks.

Distributed control networks consist of an arbitrary number of nodes (greater than 1) interconnected in an arbitrary fashion. Our discussion will concentrate on networks consisting of nodes connected by bi-directional communications paths. The interconnection topology used is chosen on the basis of reliability (usually indicating higher connectivity), bandwidth and delay (which requires traffic analysis and may require more and faster interconnection), and cost. Optimization procedures which optimize some of these factors while using constraining values of the others have been described [Frank and Chou, 1974], [Chou and Kershenbaum, 1973], [Wilcov, 1972].

General problems encountered include routing, congestion and flow control, and reliability. Routing, flow control, and congestion are discussed in other sections. Reliability is largely a function of the particular topology used, and usually must be traded off against cost, since increasing connectivity is really the only tool available. However, careful analysis often yields surprising results: in one case it was found that a very small increase in connectivity yielded a very large increase in reliability in the ARPA network [Frank and Chou, 1974]. Optimization of bandwidth and delay is also difficult. For nontrivial networks, analysis of topology is sufficiently complex that computer assistance is required (e.g., see [Frank, 1973]).

Some very important problems arise in networks without centralized control. These include resource allocation and control, security and privacy, failure detection and recovery, and synchronization. These problems largely arise because it is impractical for every node to store duplicate copies of all necessary information, and it is virtually impossible to keep such copies identical anyway. Resource allocation and synchronization require a primitive level of mutual exclusion between communicating processes. All conventional schemes are inappropriate in the presence of finite but arbitrary communications delays. Security and privacy are difficult to provide because the data sent and received is available for inspection at unpredictable places in the network (and knowledge of the presence of information is itself information). Additionally, access to protection information is subject to non-simultaneity caused by the communications delay. Withdrawal of permission may thus come after information has been leaked. See [Kleinrock, 1974] for discussions of several of these topics.

The advantages of highly interconnected distributed control networks are those that can be obtained by causing computers to communicate, plus advantages caused by the approach. The latter include reliability, survivability, and low cost [Roberts and Wessler, 1970], [Baran, 1964]. The appeal of this approach is so great that commercial networks are being built to exploit the situation [Cohen, 1974].

Special cases of the arbitrarily connected distributed control network are interesting. Davies [1971] introduces a concept called the isarithmic network, in which empty packets are passed around much as in a ring network. Flow control becomes less of a problem, but new problems are introduced. For example, it is difficult to regenerate lost empty packets in the event of a node failure or partitioning, and the location of empty packets may be wrong often enough to increase delay. Very large networks (thousands of nodes) raise many problems, including minimizing the number of hops while keeping cost low, and routing information to its destination in a near-minimal number of hops [Frank, 1973 (b)], [McQuillan, 1974], [Kleinrock, 1974]. Interconnection of

networks has become very important, particularly in Europe. The IFIPS helps sponsor a group called the Internetwork Working Group, which has produced a large collection of literature and notes. Most U.S. members have ARPA network experience. Prof. Vinton Cerf at Stanford University is currently the head of this group.

Routing

Routing is the decision process which determines the path a message follows in a network in passing from its source to its destination. Ideally, the message should make its journey in the shortest possible time. This may not be the shortest path in the sense of fewest nodes traversed (hops). The presence of higher or lower-speed lines or heavy congestion at one or more nodes may alter the optimal path. Also, the shortest path may contain a broken line or node. The routing decisions made in a node interact with those made in other nodes in a non-obvious fashion. For example, if all nodes decide to route their messages through the same node, that node may become a bottleneck and slow all traffic through it considerably.

Routing degenerates to a trivial case if only one route exists between every pair of nodes. In such a case, the routing information is pre-computed. The only updating necessary occurs if a destination becomes unreachable.

Destination addressing can be done on the basis of a fixed address, as in the ARPA network, or on the basis of a process address, as is easily done in ring, star, and broadcast topologies. Since the latter technique requires either centralized control or an examination of every message by every node, it is generally applicable only in specialized environments. Hybrid techniques which address processes regardless of location in a network with fixed node addresses, particularly in a general network, are very desirable. A combination of "most-probable-current-address", "forwarding addresses", and "where are you?" broadcasts may prove useful. (These terms are considered self-explanatory.) The authors know of no treatment of this subject in the literature, and consider it a good area for research. The following discussion is limited to routing in non-trivial situations in which destination addresses are fixed.

Under appropriate assumptions about the traffic (infinite-source Poisson arrival rate for messages, exponentially distributed message lengths, infinite storage at nodes, and totally reliable lines and nodes), an optimal deterministic routing policy can be determined. (The queuing theory terminology will be explained below.) This policy will be referred to as a static routing strategy. The primary failing of the static strategy is the reliability assumption, which makes the scheme generally unusable except for analytic purposes. The obvious solution, and the one used in, for example, the ARPA network, is an adaptive routing policy. An adaptive routing policy is one that modifies its routing decisions based on periodically updated information about the best routes to each destination. The scheme as implemented in the ARPA network uses two tables; the primary routing table contains the name of the neighboring node that reports the minimum delay to the destination, the secondary table (used because it is more responsive to line and node failures) is a minimum-hop-to-destination table. These tables are updated roughly once a second from information sent by neighboring nodes. Each node incorporates an estimate of its own contribution to the delay into the information it sends to adjacent nodes. This scheme performs nearly as well as

the static strategy. Other policies have been developed; several of these will be discussed shortly.

Adaptive routing schemes like that used in the ARPA network have two major shortcomings. The first is that they do not necessarily produce loop-free routing. Loop-free routing means that a message never visits a node more than once. If it does, the message loops in the network until some node of the loop updates its routing tables in such a way that the loop is broken. Looping packets are delayed by a variable, but potentially large, period. They also squander node capacity and bandwidth. The ultimate cause of loops is the fact that an adaptive routing strategy routes all traffic to a destination through a single neighboring node. An unfortunate timing of routing table updates can cause loops to develop. The fact that the adaptive strategies described above route all data to a destination through the same neighboring node also causes the second major shortcoming: the peak bandwidth attainable for a message entering or leaving a node is limited to the speed of one line, the one the message travels on (assuming the routing tables are not updated between parts of a message). Ideally, it should be possible for a message to split into pieces, each travelling separate, parallel paths, and arrive much faster than they would on a single line. If two lines enter a node, and half of a message comes in on each line, the effect is the same as if the message had entered on a single line operating at the sum of the individual line speeds.

Current use of the ARPA network for encoded speech transmission, which requires a high-bandwidth data flow with high tolerance for error and low tolerance for delays, has encountered several difficulties. One of these occurs because of the routing strategy: after some number of seconds of sustained high throughput, a loop sometimes develops, causing delays. The other major problem involves a packet sequencing problem that reduces effective bandwidth [Kleinrock, 1975 (b)], [Cohen, 1975], [Opderbeck and Kleinrock, 1974].

In order to produce reliable, near-optimal routing, it is apparently desirable to be able to take advantage of the multiple paths to a destination. One scheme, called the "flow deviation" method, does this. This method produces an optimal routing strategy in steady-state conditions [Fratta et al., 1973]. Other solutions have also been proposed [Frank, Kahn, and Kleinrock, 1972], [Cantor and Gerla, 1972], [Fultz, 1972]. A centralized adaptive scheme is described by Gerla [1973]; in this scheme, flow-deviation-method tables are computed at a central routing center and then sent to the nodes. Alternatively, other methods could be used to generate the tables. To guard against failure of the central node, a backup scheme based on minimum number of hops is used by Gerla.

Routing in very large networks (thousands of nodes) is a difficult problem. The tables of best route to a particular destination become unmanageably large. Schemes such as regional routing have been proposed. Regional routing involves keeping routing tables based on regions, or groups of nodes. In the simplest approach, the region is encoded as part of a node's address. Problems exist with this approach [McQuillan, 1974]. A special case of regional routing exists when several networks are connected together. If several interconnection points (gateways) exist between networks, a node must be able to select the best one to achieve optimal performance. However, this forces nodes to know about the addressing structure and, to some extent, current state of other networks. This is an undesirable situation. Routing in

large networks remains a fertile area for research [Frank, 1973 (b)], [Opderbeck and Kleinrock, 1974], [Kleinrock, 1974].

Flow Control

Flow control is the process by which the transmission of data can be controlled to prevent flooding a receiver or communications network with data. Two general methods of flow control are commonly used: controlling permission to send data, and discarding unwanted data with negative (or no) acknowledgement. The decision process used to decide when to discard data or restrict permission in a network is further divided into local and global control. Local control means that the decision process uses only local information. Global control means that the overall status of the network is (somehow) used [Opderbeck and Kleinrock, 1974]. Local control is not theoretically sufficient for restricting entry of data into a network with distributed control, since an individual node may be able to accept data while the remainder of the network cannot. Controlling permission to send means that a sender of data is permitted to send only as much data as the receiver has given him permission to send. This technique is generally applicable, but usually complex to implement [Cerf, 1974]. Discarding unwanted data (failure to positively acknowledge) is particularly useful in conjunction with a positive - acknowledgement error-control scheme (feedback-correction) [Metcalf, 1973], where failure to positively acknowledge receipt of data is already interpreted to mean the data should be retransmitted, usually after some timeout interval.

Regardless of which technique is used, the problem of overhead arises. Controlling permission to send requires that permission, in some coded form, be sent from receiver to sender. Besides introducing line overhead, this may introduce the delay involved in sending the permission into every transaction (in the worst case), which reduces the average bandwidth attainable. Appropriate strategies can help reduce such delays to a minimum [Postel, 1974]. In the case in which surplus or unwanted data is discarded, the overhead is the discarded data. Again, appropriate strategies must be chosen to keep discarded data to a minimum [Belsnes, 1974].

In any kind of network, there is always the possibility that a process will accidentally or maliciously begin uncontrolled transmission of data, possibly in violation of protocols. In the case of a star or point-to-point topology, the data can be simply discarded. However, in the case of a ring, broadcast, or distributed control network, the network could become congested and all data flow would be disastrously affected. Various methods are used to control this problem. In the British EPSS system, the total number of packets any source can send is limited to some upper bound considered "safe" [U.K. Post Office, 1974]. In the ARPA network, the number of messages in transit from a source IMP to a particular destination IMP (not host) is limited (however, see [Heart, 1974]). Similar schemes must be used to restrict entry of too much data into a broadcast or ring network. These are local techniques. The inability of local techniques to prevent congestion under all circumstances, particularly in large networks, may indicate the need for some form of global control.

The previously described problem is a pathological case of the general network flow control problem. In a network, if the average packet output rate falls below the average input rate for long enough to consume a

large part of the network's buffering capacity in some area and cause long queuing delays, overall delay increases disastrously.

Network Analysis and Modeling

In designing a communications network it is imperative to consider expected performance characteristics and cost of implementation and operation. The variety of parameters involved in network design and implementation is enormous, and the decisions that have to be made are by no means simple or obvious. It is in this context that network analysis and modeling attempts to help the researcher or designer gain insight into what is going on and what will happen as he modifies parameters in the decision process.

Much of the network analysis and modeling work has been done with reference to particular networks, especially the ARPA network. Since the work has been done for specific systems, the validity of the approach has been demonstrated by successful implementation of the designs obtained. In addition, behavior of networks has been successfully predicted and flaws in working networks have been pinpointed.

The basic techniques used for network analysis are simulation studies and queuing theory. It has been a big advantage that many basic ideas and results were already available in the queuing theory literature. Because of their importance, some of the most relevant results of queuing theory will be briefly reviewed before we discuss network modeling itself.

Queuing Theory. This area of mathematics deals with the probabilistic behavior of waiting lines, or queues. In the simplest situation, there is a single server with a single queue of customers. Arrivals at the queue occur with some probability distribution. The most common assumption is that of Poisson arrivals, meaning that the probability of an arrival in a time interval of length Δt is approximately $p\Delta t$, where p is the constant arrival rate and $1/p$ is the mean interarrival time. Once a customer reaches the head of the line, he is served. But the service time also is variable and described by a probability distribution. The most common and simplest assumption is that the service time distribution $B(t)$ is exponential, i.e. $B(t) = 1 - \exp(-bt)$, where b is the constant service rate.

One need not, of course, assume that service is on a first-come first-served basis; it could instead be last-come first-served, or some intermediate scheme based on priorities. It may be also that the service periods are limited, and after a certain length of time the customer must leave the server and go to the end of the queue. (This is the situation for computer operating systems, which dole out so-called quanta of CPU time to programs waiting in a cyclic queue.) The algorithm for deciding which customer is to be served next, and how much time he gets, is known as the queue discipline. Given the discipline, in addition to the probability distributions for service times and arrivals, a simple computation often suffices to yield the results of interest - the mean waiting time and the mean response time (waiting time plus service time.) At least this is the case for simple Poisson arrivals and exponential service times. For more complicated distributions, or more general ones for which only means and variances are specified, definitive results are not always readily available. One is able in most cases, however, to obtain information, such as an upper bound on waiting time, which comes close to the actual value in a heavy traffic situation.

At a higher level of complexity, queuing theory deals with systems or networks of waiting lines. The classic paper on this subject describes such a system as modeling a machine shop [Jackson, 1957]. Jobs arrive at the shop in the usual Poisson-type time series and flow from department to department (with preassigned probabilities), waiting in queues at each department and eventually leaving the system when they are completed. Jackson derives a simple expression for the steady-state distribution of jobs in the system. Jackson's result showed that in a sense the network could be treated as a set of independent servers. That is, the network environment affected the overall traffic at a node, but the general form of the solution was the same as if the nodes were independent.

At about the same time that Jackson's work was published, another paper appeared which was important in the application of queuing theory to networks [Burke, 1956]. Burke showed that, for a single server, if arrivals are Poisson and service times exponential, then departures are exponential. This verifies the useful hypothesis that the flow between nodes in a network is Poisson.

About ten years later, Gordon and Newell [1967(a), (b)] published a pair of now classic papers. They deal with closed systems, in which N customers pass repeatedly through a cyclic network of servers. The behavior is shown to be the same as that of an open system where the number of customers has an upper limit N . As N becomes large, it appears that the distribution of customers in the system is regulated by the server with the slowest effective service rate. They also analyze such a system in case restrictions are imposed on maximum lengths of queues. Results are limited to equilibrium behavior in two-stage systems, and in more general systems in which N is either very large or very small.

Feeling that the practical applicability of queuing theory to computer models has been limited by mathematical intractability and the concomitant oversimplifications, Kobayashi [1973 (a), (b)] has recently applied a diffusion approximation to the study of queuing networks. The idea of studying congestion in queuing systems by diffusion approximations actually goes back several years. (See, for example, [Gaver, 1968].) As is usual, Kobayashi first assumes that the system is closed and obtains an equilibrium distribution of queue lengths. However, use of the diffusion model also allow him to solve for transient (non-equilibrium) queue-length distributions.

In the paragraphs above, we have attempted to give the reader some feel for that queuing theory research which has been most applicable to the problems we are interested in - the analysis of both communications networks and multiprogrammed computer systems (considered elsewhere in this report). The reader who is interested in learning more can consult any one of many texts on queuing theory. (For example, see [Kleinrock, 1975 (a)].)

Network Models: Mathematical Analysis by Queuing Theory and Other Techniques. The main problem in carrying over general queuing theory results to message-oriented communication networks is that messages preserve their lengths when traveling through a network and hence queue lengths at the various nodes are strongly correlated. Nevertheless, the first comprehensive analysis of communication networks [Kleinrock, 1964] showed that a reasonable independence assumption again allowed decomposition of the network

and a channel-by-channel analysis. The key result of this work was an expression for the average message delay or total time that a message spends in the network. In addition, the problem of obtaining an optimal channel capacity was studied under the assumption that network topology and average traffic flow were known and fixed.

Kleinrock later became heavily involved in design and analysis problems for the ARPA network. In this work, many practical considerations had to be taken into account that did not arise in the earlier theoretical study. Among these considerations were propagation delays (due to the long distances between nodes), processing time by each IMP, and overhead traffic due to acknowledgements, error control, etc. This work is discussed in a pair of basic papers [Kleinrock, 1969], [Kleinrock, 1970], as well as in a good review [Kleinrock, 1973].

Two other aspects of network design amenable to analysis are flow control and routing algorithms. These problems are reviewed in the ARPANET context in a recent paper [Frank et al., 1972]. Flow control is necessary to avoid deadlock caused by heavy traffic congestion. The better understanding of flow obtained from analysis (as well as from simulation studies) not only has aided in the choice of IMP buffer sizes, but has also led to such flow control features as the discarding of packets (with later retransmission) when congestion is about to occur and the occasional queuing of traffic outside the net. Development of a good routing algorithm is also essential to network efficiency. As each message enters the net, a routing decision must be made. The routing problem can be stated as a multi-commodity flow problem and solved by linear programming, but this approach is too time-consuming to use as a working algorithm. Instead, a heuristic method that seems reasonable and works well in practice is therefore actually used in the ARPA network.

The problem of routing still, however, needs analysis, as does flow control. In a recent paper [Opderbeck and Kleinrock, 1974], ARPANET flow control procedures are examined and "several design oversights ... are discussed which caused or could have caused serious performance degradations or even a complete lockup." Although most of these oversights have been repaired since their discovery, Opderbeck and Kleinrock describe a complex sequence of events which could still lead to deadlock and remark that "the possibility of other, as yet undiscovered, lockups and degradations leaves one in a most uncomfortable (in fact, untenable) position."

The work of Kleinrock et al. was mainly related to the ARPA network, or networks of similar topology. Hayes and Sherman [1971] have, however, provided some analysis for a ring network (discussed above). They were principally interested in studying the time delay caused by having to wait for an empty slot on the ring in which to insert a message. Assuming Poisson arrivals, exponential distribution of idle periods, statistical independence of line idle and busy periods, and independence of interarrival times and message sizes, they arrive at an expression for average time delay. Analytical results were compared with simulation studies for various system loads and traffic patterns. The agreement was reasonably good.

A discussion of network modeling would not be complete without brief consideration of mathematical aids to choosing the basic network topology. (See [Frank et al., 1972].) First, to estimate the reliability of a

topology, the network cutsets (sets of elements whose removal breaks all communication between two or more IMPs) are determined. (This is a graph theoretical problem.) The probability of the physical failure of any cutset is readily determined from the probabilities of failure of its components. A network topology must be chosen so that the probability of the failure of any cutset is within tolerable limits. Second, assuming a reliable topology has been obtained, one may hope to improve it or even to optimize it. Optimization usually refers to getting the maximum reliable throughput for a fixed cost. A gross estimate of network throughput can be obtained from a simple diffusion model. From that point, optimization generally proceeds iteratively. A prospective improvement (involving removal and addition of links) is obtained heuristically and inserted into the network model. If calculation shows that throughput is improved, the change is kept; otherwise it is discarded. Successive changes made in this way will produce an improved network, but not generally an optimal one.

Network Models: Simulations. At some point it always becomes impossible to put all the desired features into an analytical model, and then one must resort to simulation. A routing algorithm, for example, is best tested by simulation - i.e., by running a computer program which computes the course that a randomly chosen set of messages would follow through the network and checks for bottlenecks, exorbitant delays, etc.

The most extensive simulations of networks has been done by the Network Analysis Corporation. (See [Frank and Chou; 1972, 1973] and [Frank, Frisch, and Chou, 1970].) Their work was primarily related to the ARPANET design problem and complemented the analytical analyses discussed above. In a sense they begin where the analytical results leave off. For example, they make extensive use of Kleinrock's formula for average message delay in terms of message lengths and arrival rates, propagation delays, IMP processing time, overhead, etc. Among other things, they have tested proposed capacity assignments for the network links. The procedure is to simulate message traffic, given a routing technique. Prospective overloads are then readily identified. In addition, graphs of cost vs. throughput have been obtained [Frank, Frisch, and Chou, 1970] for given average message delay times. From these graphs one may read off the maximum throughput for given cost or minimum cost for given throughput.

To give the reader a better feel for simulation techniques, we will here discuss some details of one experiment [Frank and Chou, 1974]. Random numbers selected from appropriate distributions were used to determine the arrival times of messages at the entry nodes. Other random numbers determined message destination. The progress of the messages through the network was simulated and timed. Each such experiment was repeated 100 to 200 times, or until the results seemed to be statistically significant. From trials on model networks of 12 and 18 nodes, they noted that if traffic is uniform the throughput may be only 10 to 13 per cent higher than in the case of random arrivals. They concluded from this study that the assumption of uniform flow may be a valid simplification in network analysis. Thus, we see another aspect of interaction between analysis and simulation. Not only do analytical formulas and estimates form useful inputs into simulation models, but simulations can justify simplifications that make the analysis tractable.

Introduction

A broadcast system is one in which all potential destinations of a message receive every message. Such a system may use radio, satellite-repeated radio, or even wires as a transmission medium. A large number of transmitters and receivers can operate at one time in such a system. A broadcast system with more than one transmitter is termed a multiple-access system. We are concentrating here on special cases of the general multiple-access situation. For a more general discussion, see [Schwartz and Muntner, 1973]. Here we assume that a group of receivers and transmitters communicates via a channel. The channel can be a wire or a band of radio frequencies, for example, but the important concept is that the communicating parties share the channel. With a few important exceptions, two or more transmitters cannot use the channel at one time. An attempt to do so destroys the data sent by all. Thus, the broadcast medium requires co-operation of transmitting parties.

One of the principal disadvantages of this technique is that a particular receiver retains only a fraction of the data it receives, so that a large percentage of the receiver's processor power is wasted. This can be particularly pronounced for large networks where traffic is high. However, hardware or inexpensive microprocessor interfaces could be built that remove this disadvantage. Another disadvantage of this scheme is reliability. If any transmitter breaks in such a way that it constantly remains on, the channel is effectively jammed. For broadcast radio, the jamming could come from an external source. Note that the jamming signal need not continuously disrupt a channel; one introduced error per message slot would destroy every message.

A major problem with multiple access broadcast techniques is avoiding simultaneous transmission or collision. In any environment, since simultaneous transmission is bound to occur, methods must be devised to detect and correct this situation. This topic is discussed in the Multiple Access Techniques subsection.

Broadcast techniques have many advantages. For example, because of the bursty nature of computer (particularly human-computer) traffic, the cost of a channel can be shared by a potentially large population. Since all equipment involved can be virtually identical and hence cheap (in quantity), the cost to communicate over a broadcast channel can be made very low.

Applications of Broadcast Technology

One of the primary advantages of broadcast technology is its independence from the placement of the receivers and transmitters. Even inside a building, the convenience of running a single cable and attaching to it anywhere can often greatly outweigh the problems involved. Using radio, a small, perhaps hand-held, terminal could be carried about and used without regard for the fact that the nearest telephone line or computer is miles away [Roberts, 1972]. Besides its obvious appeal to computer programmers who would be able to work at a beach, in their yard, or in bed, a portable computer terminal would find great application in field data collection,

process control supervision, point-of-sale terminals, and many more esoteric functions that the availability of such devices would foster.

Two portable packet radio terminal projects have produced prototypes: one group is at the University of Hawaii, the other is an ARPA-sponsored effort involving Stanford Research Institute, Collins Radio, and others. While these terminals are far from hand-held, they are serving to exploit current technology and discover the areas needing concentrated research. Several papers are to be delivered at the 1975 NCC in Anaheim, California on these projects. One paper [Fralick et al., 1975] describes a presently feasible terminal as consuming one watt of power for its receiver, one watt for its small computer, milliwatts (average) for the transmitter, and considerable power for the displays, which appear to be the weak point at present.

If a portable terminal were used to connect to a large computer network such as the ARPA network, the power available at one's fingertips would be enormous. The SRI group plans connection to the ARPA network soon, and experimentation with this concept should prove interesting.

When broadcast technology is coupled with communications satellites, the potential for large cost savings opens up. A satellite channel, shared by any number of ground stations, can be used in place of a sprawling ground network, or as an adjunct to one, yielding small delays regardless of distance. Projects are currently underway to exploit the potential of multiple-access satellite channels [Heart, 1974], [Cohen, 1974], [Abramson, 1973]. Techniques used for multi-access satellite channels must take the propagation delay into account. Several such techniques are described under Multiple-Access Strategies.

Current Technology

Broadcasting Media

Wire or Cable. The simplest medium that is used for broadcasting is wire. In the ETHER system [Metcalfe, 1975], a co-axial cable connects a number of minicomputers into an in-house network. The cable is completely passive (contains no amplifiers or repeaters). When one of the minicomputers wishes to send a message to another, it does so by impressing a signal onto the cable. All connected minicomputers interested in the network at that time receive the message, check it, look at the destination address, and either discard the message or assimilate it.

Since the medium is essentially zero-delay (due to the shortness of the cable), a transmitter can eliminate most simultaneous transmissions by simply listening on the line before starting transmission. If the line is busy, he waits. Using techniques based on this simple concept, Metcalfe claims line utilization of nearly 100% at a data rate in excess of one megabaud. This type of broadcast network deserves consideration for in-house networks. (Xerox has at least one patent covering aspects of this technique, and should be contacted before commercial use is planned).

Another experiment performed recently used community-antenna-television (CATV) facilities to provide computer access via a home television set [Labonte, 1973]. Though this system did not use multiple-access techniques, many CATV systems are bi-directional and would be quite suitable for

multiple-access use. Current plans apparently call for the use of multiplexers to permit multiple use of the return channel. Careful analysis of the requirements of this application should be performed, as current multiple-access technology is developed well enough to be a viable alternative to multiplexers, and might yield significant cost savings and substantially better performance.

Radio and Satellites

Radio is a natural choice for a broadcast medium. Radio digital communication equipment can be made portable, perhaps even hand-held [Roberts, 1972], [Fralick et al., 1975]. The first broadcast radio project, the ALOHA system, was begun in 1968 at the University of Hawaii [Abramson, 1973]. The ALOHA system uses a simple random-access channel for what is dubbed the classic ALOHA technique. In this scheme, remote transmitters can send packets at any time. Luck and the bursty nature of computer traffic tend to keep channel efficiency high enough to be reasonable. (Analysis of this and other techniques is done in the section on Multiple-Access Techniques.)

Radio techniques more sophisticated than the ALOHA approach may run into the serious problem of propagation delay. Any attempt to derive timing information from data received at a remote location must take the radio signal propagation delay into account. When this is not possible, alternative schemes must be used. Several such schemes are described and analyzed later.

Packet radio transceivers generally use a form of frequency-modulated pulse-code-modulation (PCM). Because of the way FM detection and demodulation is usually done, an interesting and useful effect arises. When one of two transmitters is producing a noticeably stronger signal at a receiver, the receiver tends to capture the stronger signal while ignoring the weaker one. This can help reduce packet loss by simultaneous transmission [Metcalf, 1973]. Also, since low-duty-cycle PCM is being used, there is a good chance that the pulses from two transmitters will not overlap, but will "mesh" in a non-destructive way. In such a case, the receiver could lock onto one of the signals and receive it without loss [Fralick and Barrett, 1975]. A related problem occurs naturally in areas of high radio reflectivity (e.g., in a large city). Some radio waves leaving the transmitter echo from structures not in the direct line-of-sight, arriving at the receiver late and appearing as "ghost" pulses. This is referred to as the multipath problem. The pulse spacing, and hence the ultimate bandwidth of the channel, may be limited by multipath considerations, since if the returning echos are late enough and strong enough, they may overlap the next pulse and cause data loss.

A major limitation on the portable packet radio transceiver is range. Range is limited primarily by transmitter power, receiver sensitivity, and interference at the receiver. Transmitter power cannot be made large in a portable transmitter at present because of the lagging technology in portable power sources. Consequently, the major limitation on range will probably be the mobile transmitter. In order to boost the range of mobile transmitters, repeaters will be placed at appropriate intervals in a service area. The function of a repeater is to receive packets and forward them closer to their destination. Large problems exist in performing this seemingly simple function. For example, several repeaters might receive the original transmission.

This would cause duplicates of the transmission to be present in the repeater network and ultimately appear at the destination. This problem is conveniently solved by an appropriate message sequencing technique. A more severe problem is flooding of the repeater network. If a repeater transmits each message it hears to all repeaters that can hear it, and each of those in turn does the same thing, the network would regeneratively feed back the first messages it heard until saturation was reached. Clearly, a more reasonable repeating strategy must be used (e.g., see [Kleinrock and Tobagi, 1973]). Routing and flow control in repeater networks is a fertile research area.

Multiple-Access Techniques and Analysis

This section describes some of the techniques used to allow multiple transmitters to use a channel while still obtaining a reasonable throughput. See also [Liao, 1972].

Ground Radio Techniques

The following discussions usually assume a two-channel system and a central site which uses one channel for broadcast transmission, and the other for multiple-access input.

The simplest technique is the classic ALOHA approach. In this scheme, transmitters may transmit at any time. When a receiver receives a packet with a correct checksum, indicating that the packet was not damaged by natural interference or simultaneous transmission, the packet is acknowledged. If the terminal has not heard an acknowledgement within some time-out period, it retransmits the packet. Note that in order to avoid an infinite series of simultaneous transmissions (collisions) between two transmitters, the period between retransmissions must not be the same for all transmitters. Randomly-chosen retransmission intervals, under suitable constraints, is a good approach and is used in most analyses. For this model a channel utilization of $1/2e$ or about 18% can be obtained [Abramson, 1973]. While this seems low, it should be kept in mind that most human use of computers generates at least an order of magnitude more output than input. Hence, if a group of terminals is using a central site with a broadcast output channel of the same capacity as the input channel, as in the ALOHA network, the output channel would be more likely to be the limiting factor. A drawback of this scheme is that as traffic approaches the channel capacity, delay increases and becomes unbounded above $1/2e$. The analysis assumes infinite-source users, however. If humans were using the system, they would normally wait for a reply before sending a packet (particularly if their keyboard locked until the reply arrived) [Metcalfe, 1973]. Several adaptive control procedures to allow graceful performance degradation have been proposed [Metcalfe, 1973], [Lam and Kleinrock, 1974].

A simple modification to the classical ALOHA approach is the slotted ALOHA technique. In this technique, prospective transmitters use the output channel of the central transmitter to synchronize their transmissions. Transmitters simply wait until the end of a packet transmission to begin their transmission. This reduces the probability of collision by 50%, raising channel capacity to $1/e$ or about 36%. This system becomes less useful as the propagation delay increases to a sizable fraction of the packet transmission time [Metcalfe, 1973], [Lu, 1973].

A simple variation of slotting that can be used when the terminal is operating on only one channel is carrier sense [Abramson, 1972], in which the terminal listens on its only channel. This technique also has the effect of reducing collisions, but has the disadvantage that one terminal may be "hidden" from another by some obstacle (the hidden terminal problem), making reception impossible. A possible solution [Kleinrock, 1975 (b)] is a busy channel, a second channel carrying no information other than its own presence or absence. This channel would be turned on when a receiver was receiving a packet. A busy channel sufficiently close to the main channel frequency would require little extra hardware for detection.

Satellite Techniques

The primary features of satellite, as opposed to ground, environments are: there is no central receiver, the number of stations is small, the delay from packet receipt is on the order of many packet times, and the traffic is more uniform [Metcalf, 1973]. Several schemes have been proposed to combat the relatively low channel utilization obtained with the classical ALOHA approach. We will mention only three such schemes here. Some analysis of satellite techniques has been done [Kleinrock and Lam, 1973].

The reservation ALOHA scheme introduces the idea of a frame, which is one round trip's worth of packet slots. Those slots in the previous frame that were used successfully by a particular ground station are reserved for it in the next frame. Slots that were not used by any ground station are "ALOHA slots", and anyone can attempt to use them. It should be noted that since a ground station hears all packets as they are retransmitted by the satellite, it can immediately tell if one of its own packets needs to be retransmitted. This scheme is simple and works well under heavy loading [Crowther, 1973].

A different type of reservation scheme is the interleaved reservation ALOHA scheme, in which slots are reserved ahead of time. In this scheme, each ground station broadcasts its need for slots over the channel [Roberts, 1973 (c)]. To resolve reservation conflicts, the priority reservation ALOHA scheme assigns an "owner" to each slot. The owner has priority in the event of conflicting reservation requests [Binder, 1973].

Summary and Assessment

The problems and complexity involved in interconnecting computers are easily underestimated; many problems are encountered which must be attacked and solved to make the interconnected computers cooperate. Much of what is presently known about computer networks was discovered in the course of making planned or existing networks work. Future network builders will continue to confront and solve all the same problems over and over again unless they are aware of what has already been done. The authors hope that this report is a useful start on providing a guide to previous work. This section briefly summarizes the topics presented earlier in detail and pinpoints some areas currently needing further research.

These are several media available for communicating data between two places; the means of communication are primarily cables, radio, or light. Their use for long-distance communication requires large capital outlay;

hence most computer networks will purchase communication services from common carriers or value-added carriers. Several are available and more are planned. Circuit-switched digital carriers offer error rates on the order of 10^{-5} to 10^{-7} errors per bit, essentially speed-of-light-limited delay time, and high bandwidth. Message- or packet-switched carriers offer error rates on the order of 10^{-12} errors per bit, a cross-country delay of about 1/4 second, low-to-moderate bandwidth (until demand increases), distance-independent pricing, and high availability (over 99.5%).

Message- and packet-switched computer networks are becoming a practical technology, at least in their present form. While the problems, such as those of flow control, congestion, routing, and topological design, are not completely solved, enough is known to permit practical networks to be built. However, most of the techniques in use are not easily extended to very large networks (thousands of nodes) or to novel situations such as networks of broadcast repeaters. Good techniques for large networks, broadcast networks (particularly in the presence of repeaters), and dissimilar interconnected networks are needed before prototypes can be built and made to function adequately.

There has been a large amount of work done recently on network analysis and modeling, but much more needs to be done. It is not clear, for example, how much the common statistical assumptions (e.g. Poisson arrivals, independence of interarrival time and service time, etc.) affect the results. The effects of queue discipline, including various priority schemes, have not been studied. Most analyses also assume that traffic is light; more work needs to be done on the heavy-traffic case, in particular the consequences of blocking and how they are best handled.

Routing procedures are in need of serious, systematic analysis, although as Kleinrock [1973] says, "of all those so far discussed, this problem lends itself least to analysis." Similarly, flow control is still in need of further study.

It is questionable whether queuing theory itself will serve to solve the outstanding problems. As a recent paper [Muntz, 1974] assesses the situation, queuing methods "appear to have reached their limit of applicability. The most promising area for future work is in approximate analysis techniques such as the diffusion approximation."

Broadcast techniques have the potential to provide inexpensive, rapid, and portable computer communication. Further refinements of technique may eventually yield better performance, but current technology is sufficiently well-developed to be applied in practice. We have not discussed the potential for direct-broadcast satellites, which anyone with a small dish antenna could use, nor have we mentioned geo-asynchronous satellites, which could orbit the earth at a low enough altitude to keep transmission delays small (compared to the 1/4-second round trip time associated with geosynchronous satellites). For many applications, such as personal computer terminals, broadcast techniques will require much more development. The technology of portable power sources and computer displays will also need more development before the Dick Tracy wrist TV can become a computer terminal.

References

Abramson, N.

. 1972 "ARPANET Satellite System", ASS Note #2, NIC #11288.

1973 (a) "The ALOHA System" in Computer Communication Networks, Norman Abramson and Franklin Kuo, eds., Prentice-Hall.

(b) "Packet Switching with Satellites", AFIPS Conf. Proc. 42, p. 695.

Baran, P.

1964 "On Distributed Communications: Summary Overview", Rand Corporation Memorandum RM-3767-PR, Rand Corporation, Santa Monica, Ca.

(This document summarizes a series of 11 Rand documents. The remainder of the series is: RM-3420-PR, RM-3103-PR, RM-3578-PR, RM-3638-PR, RM-3097-PR, RM-3762-PR, RM-3763-PR, RM-3764-PR, RM-3765-PR, RM-3766-PR. This list is duplicated, with abstracts, in each document.)

Belsnes, D.

1974 "Flow Control in Packet Switching Networks", SU-DSL Technical Note #50, available as Internetwork Working Group Note #63, Oct. 1974.

Binder, R.

1973 "Another ALOHA Satellite Protocol", presented at the Sixth Hawaii Int'l Conf. on System Sciences, Jan. 1973. (Indirect Reference)

Bouknight, W.J., Grossman, G.R. and Grothe, D.M.

1973 "The ARPA Network Terminal System, A New Approach to Network Access", in Data Networks, Analysis and Design, Proceedings of the 3rd Data Communications Symp., IEEE, pp. 73-79.

Burke, P.

1956 "The Output of a Queuing System", Operations Research 4, pp. 699-704.

Cantor, D.G. and Gerla, M.

1972 "The Optimal Routing of Messages in a Computer Network via Mathematical Programming", IEEE Comp. Con. 72, pp. 167-170.

Carter, W.C.

1973 "Reliability Techniques Applicable to Message Processors", in Data Networks, Analysis and Design, Proceedings of the 3rd Data Communications Symp., IEEE, pp. 157-158.

Cerf, V.

1974 "An Assessment of ARPANET Protocols", RFC #635, NIC #30469, Apr. 1974.

Cerf, V. and Kahn, R.

1974 IEEE Transactions on Computing COM-22, pp. 637-648.

Chou, W. and Kershenbaum, A.

- 1973 "A Unified Algorithm for Designing Multidrop Teleprocessing Networks" in Data Networks, Analysis and Design, Proceedings of the 3rd Data Communications Symp., IEEE, pp. 148-156.

Cohen, D.

- 1975 Personal Communication, Jan. 1975, University of California, Information Sciences Institute, Marina Del Ray, Ca.

Cohen, I.

- 1974 "U.S. Value Added Carriers", Draft report, available as Internetwork Working Group Note #68, Oct. 1974.

Crowther, W., et al.

- 1973 "A System for Broadcast Communication: Reservation - ALOHA", Proceedings of the Sixth Hawaii Int'l Conf. on System Sciences, Jan. 1973. (Indirect Reference)

Crowther, W., Heart, F., McKenzie, A., McQuillan, J. and Walden, D.

- 1974 "Report on Network Design Issues", Bolt, Beranek, and Newman Report 2918. Also available as Internetwork Working Group Note #64, Oct. 1974.

Crowther, W., McQuillan, J. and Walden, D.

- 1973 "Reliability Issues in the ARPA Network", in Data Networks, Analysis and Design, Proceedings of the 3rd Data Communications Symp., IEEE, pp. 159-160.

DATAPAC

- 1974 Datapac Standard Network Access Protocol, the Computer Communications Group of Trans-Canada Telephone System.

Davies, D.W.

- 1971 "The Control of Congestion in Packet Switching Networks", ACM/IEEE Second Symp. on Problems in the Optimization of Data Communications Systems, Oct. 1971, IEEE, pp. 46-49.

Davies, D.W. and Barber, D.L.A.

- 1973 Communication Networks for Computers, Wiley Series in Computing, John Wiley and Sons.

Farber, D.

- 1972 "Data Ring Oriented Computer Networks", in Computer Networks, R. Rustin, ed., Prentice-Hall, pp. 73-93.

Farber, D. and Larsen, K.

- 1972 "The System Architecture of the Distributed Computer System--the Communications System", in Computer Communications Networks and Teletraffic, J. Fox, ed., pp. 21-27.

Farmer, W.D. and Newhall, E.E.

- 1969 "An Experimental Distributed Switching System to Handle Bursty Computer Traffic", Proc. ACM Symp., Pine Mountain, Ga., Oct. 1969. (Indirect Reference)

- Fralick, S.C., Brandin, D.H., Kuo, F.F. and Harrison, C.
 1975 "Digital Terminals for Packet Broadcasting", draft report, to be presented at the National Computer Conf., 1975.
- Fralick, S.C. and Garrett, J.C.
 1975 "Technological Considerations for Packet Radio Networks", draft report, to be presented at the National Computer Conf., 1975.
- Frank, H.
 1972 "Optimal Design of Computer Networks", in Computer Networks, R. Rustin, ed., Prentice-Hall, pp. 167-183.
- 1973 (a) "Providing Reliable Networks with Unreliable Components", in Data Networks, Analysis and Design, Proceedings of the 3rd Data Communications Symp., IEEE, pp. 148-156.
- (b) "The Practical Impact of the Recent Computer Advances on the Analysis and Design of Large Scale Networks", First Semiannual Report to Advanced Research Projects Agency, Network Analysis Corporation, May 1973. Available from National Technical Information Service.
- Frank, H. and Chou, W.
 1972 "Topological Optimization of Computer Networks", Proc. IEEE, Vol. 60, No. 11, pp. 1385-1397.
- 1974 "Network Properties of the ARPA Computer Network", Networks 4, pp. 213-234.
- Frank, H., Frisch, I.T. and Chou, W.
 1970 "Topological Considerations in the Design of the ARPA Computer Network", AFIPS 36, pp. 581-587.
- Frank, H., Kahn, R.E. and Kleinrock, L.
 1972 "Computer Communication Network Design - Experience with Theory and Practice", AFIPS (SJCC), pp. 255-270.
- Fratta, L., Gerla, M. and Kleinrock, L.
 1973 "The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design", Networks 3, pp. 97-133.
- Fultz, G.L.
 1972 "Adaptive Routing Techniques for Message Switching Computer - Communication Networks", UCLA Computer Science Department, UCLA-ENG-7252, July 1972. (Indirect Reference)
- Gaver, D.P., Jr.
 1968 "Diffusion Approximations and Models for Certain Congestion Problems", Journal of Applied Probability 5, pp. 607-623.
- Gerla, M.
 1973 "Deterministic and Adaptive Routing Policies in Packet-Switched Computer Networks", in Data Networks, Analysis and Design, Proceedings of the 3rd Data Communications Symp., IEEE, pp. 23-28.

Gordon, W.J. and Newell, G.F.

1967 (a) "Closed Queuing Systems with Exponential Servers", Operations Research 15, pp. 254-265.

(b) "Cyclic Queuing Systems with Restricted Queue Lengths", Operations Research 15, pp. 266-276.

Hansler, E.

1974 "A Heuristic Configuration Procedure for Cost Minimal Communication Networks", IBM Research Document RZ-666 (#22402), Oct. 1974.

Hassing, T.E., Hampton, R.M., Bailey, G.W. and Gardella, R.S.

1973 "A Loop Network for General Purpose Data Communications in a Heterogeneous World", in Data Networks, Analysis and Design, Proceedings of the 3rd Data Communications Symp., IEEE, pp. 88-96.

Hayes, J.F. and Sherman, D.N.

1971 (a) "Traffic and Delay in a Circular Data Network", ACM/IEEE Second Symp. on Problems in the Optimization of Data Communications Systems Oct. 1971, IEEE, pp. 102-107.

(b) "Traffic Analysis of a Ring Switched Data Transmission System", Bell System Technical Journal 50, pp. 2947-2978.

Heart, F.E.

1974 "Interface Message Processors for the ARPA Network", Bolt, Beranek, and Newman Report 2852, Bolt, Beranek, and Newman Inc., Cambridge, Mass.

Heart, F.E., Kahn, R.E., Ornstein, S.M., Crowther, W.R. and Walden, D.C.

1970 "The Interface Message Processor for the ARPA Computer Net", Proc. SJCC, AFIPS, pp. 551-567.

Heart, F.E., Ornstein, S.M., Crowther, W.R. and Barker, W.B.

1973 "A New Minicomputer/Multiprocessor for the ARPA Network", Proc. National Computer Conf., AFIPS, pp. 529-537.

Itoh, K and Kato, T.

1973 "An Analysis of Traffic Handling Capacity of Packet Switched and Circuit Switched Networks", in Data Networks, Analysis and Design, proceedings of the 3rd Data Communications Symp., IEEE, pp. 29-37.

Jackson, J.R.

1957 "Networks of Waiting Lines", Operations Research 3, pp. 518-521.

Kanodia, R.K.

1974 "A Lost Message Detection and Recovery Protocol", RFC #663, NIC #31387, Nov. 1974.

Kaye, A.R.

1972 "Analysis of a Distributed Control Loop for Data Transmission", proceedings of the Symp. on Computer Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, Apr. 1972.

Kleinrock L.

1964 Communication Nets: Stochastic Flow and Delay, McGraw-Hill.
(Reprint: Dover, 1972).

1969 "Models for Computer Networks", Proc. Int'l. Conf. Communs.,
pp. 21-9 - 21-16.

1970 "Analysis and Simulation Methods in Computer Network Design", AFIPS
(SJCC), pp. 569-579.

1973 "Scheduling, Queueing, and Delays in Time-shared Systems and
Computer Networks", in Computer-Communication Networks, N. Abramson
and F. Kuo, eds., Prentice-Hall, pp. 95-141.

1974 "Research Areas in Computer Communications", Computer Communication
Review, SIGCOMM Quarterly Review, W. Chu, ed., 4, pp. 1-4.

1975 (a) Queueing Systems. Vol. 1: Theory, John Wiley

(b) Personal Communication, Jan. 1975, University of California
at Los Angeles.

Kleinrock, L. and Lam, S.S.

1973 "Packet-switching in a Slotted Aloha Channel", AFIPS Conf. Proc.
42, p. 793.

Kleinrock, L., Naylor, W.E. and Opderbeck, H.

1974 "A Study of Line Overhead in the ARPANET", available as Internetwork
Working Group Note #71, to appear in Communications of the ACM.

Kleinrock, L. and Tobagi, F.

1973 "Routing in Packet Radio System - Controlled Flooding Using Handover
Numbers", Packet Radio Temporary Note #11, NIC document #13645, Jan.
1973.

Kobayashi, H.

1973 (a) "Application of the Diffusion Approximation to Queuing Networks:
Part I - Equilibrium Queue Distributions", First Annual SIGMETE Symp.
on Measurement and Evaluation, ACM, pp. 54-62.

(b) "Applications of the Diffusion Approximation to Queuing Networks:
Part II", Proc. Seventh Ann. Princeton Conf. on Inform. Sci. and
Systems, pp. 448-453.

Labonte, R.C.

1973 "A General Purpose Digital Communications System for Operation on
a Conventional CATV System", IEEE Comp. Conf. 73, pp. 85-88.

Lam, S.S. and Kleinrock, L.

1974 "Packet Switching in a Multi-Access Broadcast Channel: Dynamic
Control Procedures", IBM Research Document RC-5062 (#22361), Oct.
1974.

Liao, H.H.J.

- 1972 "Multiple Access Channels", Ph.D. thesis, Technical Report A72-2, and ALOHA System, University of Hawaii, Sept. 1972. (Indirect Reference)

Loomis, D.

- 1973 "Ring Communication Protocols", Technical Report #26, Dept. of Information and Computer Science, University of California at Irvine, Jan. 1973.

Lu, S.

- 1973 "Dynamic Analysis of Slotted ALOHA with Blocking", ASS Note #36, NIC #14790, Mar. 1973.

Lucky, R.W.

- 1973 "Common-Carrier Data Communication", in Computer Communication Networks, Norman Abramson and Franklin Kuo, eds., Prentice-Hall, pp. 142-196.

Martel, C.C., Cunningham, I.M. and Grushcow, M.S.

- 1974 "The BNR Network: A Canadian Experience with Packet Switching Technology", in Computer Networks, 2, IFIPS, North-Holland Publishing Company, Amsterdam.

McQuillan, J.M.

- 1974 "Adaptive Routing Algorithms for Distributed Computer Networks", Ph.D. Thesis, also Bolt, Beranek, and Newman Report #2831, Bolt, Beranek, and Newman, Inc., Cambridge, Mass. (Indirect Reference)

McQuillan, J., Crowther, W., Cosell, B., Walden, D. and Heart, F.

- 1972 "Improvements in the Design and Performance of the ARPA Network", Proc. FJCC, AFIPS, pp. 741-754.

Metcalf, R.M.

- 1973 "Packet Communication", MAC TR-114, Project MAC, Massachusetts Institute of Technology.

- 1975 Personal Communication, Jan. 1975, Xerox Palo Alto Research Center, Palo Alto, Ca.

Mimno, N.W., Cosell, B.P., Walden, D.C., Butterfield, S.C. and Levin, J.B.

- 1973 "Terminal Access to the ARPA Network: Experience and Improvements", IEEE Compcon 73, pp. 39-43.

Muntz, R.R.

- 1974 "Analytic Models for Computer System Performance Analysis", UCLA Computer Science Dept. Quarterly 2, pp. 49-66.

Opderbeck, H.

- 1975 Personal Communication, Jan. 1975, University of California at Los Angeles.

Opderbeck, H. and Kleinrock, L.

- 1974 "The Influence of Control Procedures on the Performance of Packet-Switched Networks", presented at the Nat'l. Telecommunications Conf. in San Diego, Dec. 1974.

- Pierce, J.R.
 1971 "Network for Block Switching of Data", IEEE Conv. Rec., New York, N.Y., Mar. 1971. (Indirect Reference)
- Postel, J.B.
 1974 "Survey of Network Control Programs in the ARPA Computer Network", Mitre Technical Report 6722, Rev. 1, Mitre Corporation, McLean, Va., Oct. 1974.
- Pouzin, L.
 1973 (a) "Network Architecture and Components", presented at the First European Workshop on Computer Networks, Arles, France, 1973.
 (b) "CIGALE, the Packet Switching Machine of the CYCLADES Computer Network", Reseau Cyclades, Report MIT 556, available from the author.
- Roberts, L.G.
 1972 "Extensions of Packet Communication Technology to a Hand Held Personal Terminal", Proc. of the SJCC, AFIPS, pp. 295-298.
 1973 "Interleaved Satellite Reservation System", AFIPS Conf. Proc. 42, p. 711.
- Roberts, L.G. and Wessler, B.D.
 1970 "Computer Network Development to Achieve Resource Sharing", Proc. SJCC, AFIPS, pp. 543-549.
- Schwartz, J.W. and Munther, M.
 1973 "Multiple-Access Communications for Computer Nets", in Computer Communication Networks, Norman Abramson and Franklin Kuo, eds., Prentice-Hall.
- Spragins, J.D.
 1972 "Loops Used for Data Collection", presented at the Symp. on Computer - Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, Apr. 4-6, 1972, pp. 59-75.
- UK Post Office
 1974 "Some Considerations on Flow Control for International Packet Transport", available as Internetwork Working Group Note #8, Jun 1974.
- Wilkov, R.S.
 1972 "Design of Computer Networks Based on a New Reliability Measure", presented at the Symp. on Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, Apr. 4-6, 1972, pp. 371-384.
- Yuen, M.L.T., Black, B.A., Newhall, E.E. and Venetsanopoulos, A.N.
 1972 "Traffic Flow in a Distributed Loop Switching System", Proc. of the Symp. on Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, Apr. 1972.

Introduction

The first topics of this section, Multiple Copies of Data Bases in a Distributed Environment, Naming and Locating Resources, Mutual Exclusion, and Deadlock, are intended to sensitize the reader to some of the difference between distributed and centralized computer systems. They do not necessarily describe all aspects of the problems, but serve primarily as background for the last section, Accessing Remote Resources - Protocols, which describes the techniques used to achieve communication and control objectives in a distributed environment.

This section is concerned with the mechanisms used to access and allocate resources to permit resource sharing in a distributed network [Roberts and Wessler, 1970].

Multiple Copies of Data Bases in a Distributed Environment

The problems of maintaining duplicate copies of data bases are certainly not unique to computer networks. However, computer networks require the routine use of duplicated data bases in an environment where the communications delay between copies is a sizable part of a second, and the probability of failure or unavailability of one or more of the copies may be high. Multiple copies of such data as resource status tables, file directory or catalog information, user authentication data, and user accounting information may be integral to the operation of a network. So their maintenance assumes major importance.

The general problem of maintaining duplicate copies in a network includes several important sub-problems. We will describe several of these problems in detail before attempting to describe the general problem further.

Multiple-User Problems. When several processes must update a data base, steps must be taken to insure that the results are consistent and dependent on the order, but not the timing, of operations. (The latter requirement helps insure repeatability). Some type of mutual exclusion (q.v.) is generally necessary to keep more than one process from updating the data base at the same time, which can lead to inconsistencies when the updates overlap but do not coincide. Mutual exclusion is generally implemented by a data base locking mechanism. Such mechanisms introduce the problem of what to do when a process which has locked the data base dies (possibly leaving it in an inconsistent state) or simply keeps it locked for a long time. Solutions to this problem are highly application-dependent.

Multiple Data Base Problems. If multiple data bases are being maintained, the order of applying updates to the copies becomes important. If it is crucial that one of the copies always be correct, then it must be the first to receive each update. This creates a reliability problem because of the trust placed in one particular copy. Mechanisms for changing

over to a different copy in the event of a failure are necessary. On the other hand, if there is no need for some copy to always be correct, updates can be made to any copy [Johnson and Thomas, 1975], and then distributed to the others.

Unreliability and the Reconciliation of Copies. In an unreliable environment, it is inevitable that one or more copies of a data base will be out of service occasionally and miss updates or, for some other reason, diverge from the correct copies. An even worse problem is caused by partitioning, the division of a network into two or more non-communicating parts. The copies in the separated parts of the network can become irreconcilably different if the data base is modified on the basis of its contents. Some reconciliation scheme must be available, where reconciliation is possible, to make the copies identical again. Simply making a copy of some copy of the file which has been deemed "correct" (by some method) will work for small files, but is impractical for large ones. Some scheme of audit trails, etc., will normally be necessary. A scheme to eliminate duplicate updates to a file has been proposed by Johnson and Thomas [1975]. This scheme uses the monotonicity of time at the updating sites, and places an ordering on the sites where necessary, to decide which of several conflicting updates to believe (cf. Integrity section).

The Overall Problem. The above discussions illustrate a few of the problems of maintaining duplicate copies of data bases in a network environment. The relatively long delays encountered in networks greatly reduce the amount of interaction that processes can do without creating impractically slow systems. At the present level of understanding of this problem, the best solutions to be expected are specific ones that work in a useful percentage of the cases encountered in practice, or even just in some particular case of sufficient importance.

Naming and Locating Resources

Before a resource can be used by some process (any of the current definitions of process will suffice in this context), the process must have some way of uniquely referring to the resource. This requires that resources be given unique names. Once the name of a resource is known, the process must determine where the resource is located. (In some environments, such as ring or broadcast networks, the process may never need to know where the resource is. We will concentrate on environments where this is not the case.) This is the resource locating problem.

Name Spaces. Some entities, e.g., processes and data files, are frequently created and destroyed. Each newly created entity must be given a unique name that serves to distinguish it from all other entities. Names are often logically grouped together in such a way that all names within the group are guaranteed unique, though they may coincide with names in other groups. Such a group of unique names is often referred to as a name space. An example is the set of names of processes on a particular processor. Another example is the set of file names on a volume of files with one directory. Two approaches to generating unique names in a name space can be taken. One is to generate the names in a unique place. The name assigner can thus insure uniqueness by checking for previous assignment of the name. In distributed control networks, this scheme avoids many problems, but creates two of its own - a bottleneck and a reliability problem. The second approach is to partition the name space so that the names generated in different places are still guaranteed unique. This can be seen as an abstraction which encompasses the concept of directories and catalogs (see below). Farber and Larson [1972 (b)] give a description of the process naming scheme used in the UCI DCS (University of California at Irvine, Distributed Computer System) which uses a partitioning of the name space.

Note that care must be taken when naming objects to avoid giving two entities the same name. An outgrowth of this problem occurs when the name of an entity is stored someplace inaccessible or unknown to the name assigner, and the entity is destroyed. If the name is reused, a reference to the old entity name will, in fact, refer to the new entity. For this reason, it might be desirable to have a large enough name space that names are never reused [Saltzer, 1974].

Directories and Catalogs. Resources are frequently hierarchically named. That is, each name in the composite name makes the reference more specific. For example, a data file may be named by concatenating the generic class of device on which it resides, such as DISK, a particular device or volume, such as MASTER3, a group of files owned by a particular user, such as SCHWEMEIN, and a particular file, such as MAILBOX. This yields (using "/" as a separator): DISK/MASTER3/SCHWEMEIN/MAILBOX. (Note that DISK/MASTER3/ SCHWEMEIN refers to a set of files and DISK/MASTER3 refers to an entire disk volume.) This scheme uniquely describes a resource's location, hence solves the resource location problem trivially. However, it has been found in practice to be inconvenient for users of a resource to have to know its location, since locations may often change.

The hierarchical scheme is readily extended to include a name to identify the site in a network (and even which network) where the resource can be found; e.g., if the file of our previous example is located at site ILLINOIS-CAC and can be accessed via the ARPA network, a possible name for the file would be ARPANET/ILLINOIS-CAC/DISK/MASTER3/SCHWEMEIN/MAILBOX. A variation of this scheme has been proposed for the ARPANET [Crocker, 1974]. Any composite name which directs a search for the object it names is called a pathname, since it defines a "path" through the hierarchy of names to a particular object. Each step of the search involves looking up the next sub-name in a directory of names to narrow the search. Each directory, except the last one, will direct the search to another directory. Another case of hierarchical naming is also often seen, although not usually in the network context. In this technique, names are formed by the concatenation of directory names, as before, but the directory names do not necessarily correspond in any meaningful way to real objects, devices, or locations. Each directory could be on a different device, or even at different sites on a network. The only requirement is that each directory be able to direct the search to the next one or to the resource itself. This scheme is more convenient for general use, as the name of a resource need never change, even if it is moved.

Several problems arise when the general hierarchical naming scheme is applied in a distributed environment. If the hierarchy spans two or more computers, problems may arise when one or more of them is out of service. Since the location that is down may contain the only directory that can direct the search to locations that are not down, available resources may not be accessible because they cannot be located. Obvious solutions include keeping multiple copies of parts of the hierarchy and not spreading the directories across devices or computers. These solutions are difficult or inconvenient for some environments.

Another approach to the problems of naming and locating resources is commonly referred to as a catalog. A catalog is much like a directory in that it contains location information about resources. However, a catalog is used simply to reduce the effort of finding a resource - it is not essential. The catalog approach has the advantage that resources can always be found if they are available.

The UCI DCS file system [Farber and Heinrich, 1972] uses a one-level (non-hierarchical) naming scheme. To increase the probability that a file can be found in spite of failures, several "central component" processes exist. These are identical and serve only to associate user names with catalog names. There is one catalog that contains the "volume process" names associated with all files belonging to a particular user. If this catalog is unavailable, an exhaustive search of all volumes can still locate any needed file. The "volume process" that searches a particular volume would reside on the same computer as that volume [Farber and Heinrich, 1972]. This approach would require modification for use in large networks because of the cost of searching every volume in the network for a file, in case some catalog is unavailable. If a number of users initiated volume searches at the same time, the network might be able to do nothing else.

General hierarchical directory schemes, as well as catalog schemes, could benefit from duplicate copies of critical data on different machines. This is a recurrence of the familiar multiple-copies-of-data-bases problem. Its solution for the special case of maintaining directories for locating resources would allow efficient and reliable resource locating, and allow easier distribution of resources in a network. Little treatment of this problem is available in the literature, and further research is needed [Johnson and Thomas, 1975], [Chu, 1975], [Postel, 1975 (b)].

Naming in a Network. In a network of heterogeneous computers, it is expected that each computer will have its own naming schemes for objects. A network user should not have to know the naming schemes in use at every computer site whose resources he wants to use. Hence, a motivation exists to standardize the naming structure of every computer in the network. Since standardization is virtually impossible, an alternative is to provide a network-standard canonical form for names which all names can be uniquely mapped to and from. This approach is a reasonable first step toward uniform naming [Crocker, 1974].

Mutual Exclusion

Mutual exclusion is generally taken to be the function of keeping two or more processes from entering critical sections at the same time. A critical section is a section of a program that performs a series of inseparable actions. A good example of a critical section of a program is the code that enters items into linked list. If more than one program was allowed to enter items at the same time, the linked list could be left in an inconsistent state. This problem has been extensively studied in recent years. (For a good tutorial, see [Shaw, 1974]; also [Knott, 1974].) However, most of the current work has assumed a shared memory among the one or more processors involved. This makes it possible to use the inherently single-user property of the memory to achieve a primitive level of mutual exclusion, which can be built upon to form more useful primitive operations. The classic paper in this area was written by Dijkstra [1968], who describes a pair of synchronization primitives known as "P" and "V". Other primitives have been described (e.g., see [Knott, 1974], [Wodon, 1972], [Kosaraju, 1973] [Lamport, 1974], [Patil, 1971]); virtually all are adequate for achieving mutual exclusion in practice (with varying degrees of convenience and efficiency). Mutual exclusion is complicated in an environment without shared memory because the primitive level of exclusion that had been provided by the memory hardware is no longer available [Knott, 1974].

The need for mutual exclusion arises in many facets of process interaction. When processes wish to share state information of any kind, e.g., file status information, mutual exclusion is usually needed to prevent near-simultaneous accessing and modification of the state information from leaving the state information in an inconsistent state.

In some environments, all references to shared data can be performed by one process (or set of synchronized processes) which simply performs requested operations for other processes, one at a time. This

process essentially becomes a subroutine of any process whose request it services, taking the place of a critical section, and providing exclusion by virtue of the "one-at-a-time" restriction. This technique is often used in data base and file management schemes and has been used recently as the basis for at least two network operating systems [Retz et al., 1974], [Bouknight et al., 1973]. However, many operations performed on data require that it first be examined, then modified on the basis of the examination. In order to use the one-process technique, a facility to "lock" the data, thereby preventing other processes' requests from being serviced, must be provided.

The one-process scheme has several disadvantages. A major one is the delay that can arise when a process locks a data base for a long period of time. One way of preventing this period of time from exceeding some upper bound is to time out the lock; after the data has been locked for some amount of time, the manager process unlocks it and notifies the locking process. This solution is fraught with peril, however, as the data may have been left in an inconsistent state. Some recovery mechanism, if practical, would have to be an integral part of the scheme. Another major problem is the delay that can arise simply because of the number of requests the process receives and the fact that, in general, the process cannot overlap processing of requests to take advantage of any parallelism possible. A very important problem with this scheme is reliability. If all processes using the single-process exclusion scheme must depend on a lone process for service, some means of providing a replacement in the event of failure must be found. This, unfortunately, degenerates to the "multiple copies of data bases" (q.v.) problem.

Deadlock

Deadlock occurs when a process is blocked from further execution because it must wait for some event to occur or for some condition to be satisfied, but the event will never occur. This intuitive definition encompasses deadly embrace deadlocks, in which each of two or more processes is waiting for an event to occur that can only occur if one of the other processes proceeds, and effective deadlock, which occurs when a process is waiting for a logically possible condition that will nevertheless never occur. We will discuss only the first type in detail. Deadlocks are well described in the literature but practical treatment is generally limited to non-distributed environments. (For a good treatment, see [Shaw, 1974.]) Effective deadlocks can be caused by process malfunction. These problems are sometimes easy to find, since the processes which deadlock stay that way long enough for the system to discover why they stopped. Some systems can provide assistance in recovery from this situation by informing processes when a communicant of theirs has died. This problem has not been given much theoretical treatment, but is addressed to some degree by most operating systems. Another way to cause an effective deadlock is through the normal action of a resource scheduler [Holt, 1971]. For example, some process might be ready to run in every respect; however, the particular decision algorithm causes the processor scheduler to continually run other processes in preference. Effective deadlock may occur in many situations when the progress of a process depends on random variables, such as

arbitrary scheduling choices or the simultaneous availability of some set of resources. (This is also known as starvation [Dijkstra, 1968], [Knuth, 1966].)

Consider the following situation: two processes, P1 and P2, each will require the use of resources A and B during their execution. Suppose P1 begins execution and immediately obtains resource A, and P2 begins execution and immediately obtains resource B. At some point, P1 needs resource B and attempts to obtain it. P1 is suspended and cannot continue to completion until it obtains B. At some point, P2 needs resource A and, in attempting to get it, is also suspended. Now, both processes are blocked, waiting for the other process to complete and free its resources. This is a classic deadly embrace deadlock. A deadlock prevention scheme would have prevented this situation from developing. After it has happened, a deadlock detection scheme can be used to identify the deadlock and initiate deadlock recovery operations.

Deadlock Prevention. Prevention of deadlock in a centralized computer system can be performed in several ways. The most common is preallocation, where all resources that a process will need are obtained before the process is started. If all the resources are not available, the initiation of the process waits until they are. If the preallocation is being performed in a centralized resource allocator, this scheme is simple and easy to implement. One scheme that a process already running can use to obtain resources without using a centralized resource allocator is to make multiple claims for resources; that is, request several resources in an inseparable request. For example, the "multiple-P" operation of Patil [1971] allows this. However, since a process can safely make only one claim, this is virtually equivalent to a central allocator. It is also possible to make requests for individual resources at any time without fear of deadlock, if an order is placed on resources and they are requested in monotonic order [Shaw, 1974], [Havender, 1968]. The resource requests can come at any time in the life of a process, provided the ordering is observed. Schemes based on these ideas have been described for distributed networks [Somia, 1973], [Chu and Ohlmacher, 1974]. Unfortunately, preallocation or ordered allocation is not always possible in practice, since the resources needed are not always known in advance. Furthermore, some resources may be underutilized since processes may not actually need all the resources until late in execution. Deadlock prevention, because of its expense, is probably best used by processes that cannot be restarted once they begin, processes that utilize all their resources, or processes (such as internal processes belonging to an operating system) which must always be available.

Deadlock Detection and Recovery. When a deadlock occurs, some action must be taken to free up the resources being consumed by the halted processes. In fact, the ideal situation would be to allow as many of the processes as possible to somehow continue. However, the deadlock must first be detected, and the involved processes identified. In a centralized scheme (the important feature being shared memory), appropriate data structures can be used for representing process ownership of resources and resource ownership by processes. At the point that a process is blocked when its attempt to allocate fails (or at some other appropriate time), the data structure can be searched and any deadlocks,

as well as the identities of the processes involved, can be easily found [T.J. Miller, 1974], [Shaw, 1974]. In a distributed environment (i.e., no shared memory), it is still relatively easy to detect deadlock, as long as the different systems provide uniform names for use in following through their resource and process data structures.

Once a deadlock has been identified, action must be taken to free some of the resources being tied up. Two approaches to this function are resource preemption and process termination. Resource preemption means that resources are deallocated from processes in the deadlock and given to another process so it can continue. The preempted process then has a request for the resource entered in its behalf. It is necessary in this scheme for the resource to be in a reusable state when it is preempted and when it is returned after use. This is not always possible. Process termination means that some process in the deadlock is forcibly terminated, presumably to be restarted later, and its resources freed. This is a more drastic solution than resource preemption, but is usually much easier. This scheme also requires that the resources belonging to the process be in a reusable state.

The process (or processes) which is delayed by preemption or terminated for later restarting can be chosen randomly or by some suitable set of criteria. Possible criteria might be the amount of money the process has cost so far, deadlines that must be met, whether a human is waiting on the process, or whether the process can be restarted successfully. (Programs which modify data bases, for example, often cannot [Shaw, 1974]).

Accessing Remote Resources-Protocols

Introduction. In a network environment, the problem of accessing resources is totally different from what it is in the traditional computer system, in part because of the absence of a common store. To solve this problem, protocols are developed to allow the communicating computers to have a common basis for interaction and control. Some of the relevant issues in protocol design that are discussed in this section are error control, flow control, resiliency, and synchronization.

Protocol layering (i.e., the functional isolation of different levels of network interaction) has proved to be a very useful concept. The organization of most of this section will be based on one such possible layering. Starting with the lower levels we will discuss the protocols for establishing connections (host-host protocols), and then move on to telecommunication protocols, load sharing, and remote job entry protocols. We will then consider the tools for the formal analysis and definition of protocols, and finally consider the concept of a distributed operating system.

Establishing Communications. The lowest level protocol in most computer networks is that which establishes communications between the computers in the network. This protocol, which we will refer to as "host-host", then serves as a basis for more specialized protocols. In this section we will discuss three basic host-host protocols. These will illustrate the basic form of a host-host protocol and the major points it must address.

The ARPA host-host protocol is based on a line-switched discipline. The concept of a socket is introduced as an intermediate name space for establishing associations between connections and the processes in the machine. One of the problems with this protocol is that it is not sufficiently resilient to be able to recover from the loss of messages in the communications subnet. There has been much discussion of solutions to this problem [Burchfiel and Tomilson, 1973] [Meyer, 1973] [Hathaway, 1973] [Walden, 1973], but to date the most reasonable solution requires significant extensions to the present protocol [Kanodia, 1974]. This solution may be impractical since an implementation of the protocol already requires a fairly substantial programming effort. A survey of the implementations of this protocol [Postel, 1974 (c)] indicates that the programming effort is severely compounded for systems without adequate interprocess communication systems. This constraint may be inherent in the requirements of networking protocols.

Early in the history of the ARPA network another host-host protocol was proposed by Bressler et al. [1972] based on a message switched discipline and an interprocess communication system proposed by Walden [1972]. This protocol suffers the same resiliency difficulties as the official ARPA host-host protocol. However, it may be more efficient to implement. Bressler had implemented a version of the protocol based on a similarly structured interprocess communication system for the PDP-10 and found the protocol implementation to be "something like an order of magnitude smaller" than the protocol implementation for the official protocol on the same machine. However, this may have been due primarily to the good "impedance match" of the two interprocess communication facilities; i.e., a message switch protocol interfaces well to a message-switch-oriented interprocess communication system.

Recently a new host-host level protocol has been proposed by Cerf et al. [1974] as a result of the deliberations of the Inter-Network Working Group. This protocol is a Sequencing Positive ACK/Retransmission (SPAR) protocol and is resilient. Furthermore, implementation will probably not require any more system resources than the official protocol and perhaps less for some systems. Also, this protocol provides a much better flow control scheme than the official protocol. In the official protocol the destination sends separate control messages, called Allocates, which adjust the number of messages and bits the source can send. Some allocation strategies can easily limit the effective bandwidth of the channel because the source is waiting for notification to send or simply because of the bandwidth required for the allocation messages. The Inter-Net protocol uses a moving window to describe the number of bits that can be sent. In addition, the commands from the destination which move the edge of the window are a standard part of messages carrying data in the other direction. The moving window concept is not only the basis for flow control and for lost-or-duplicate message detection, but it also allows hosts to reassemble out-of-order messages. It has been argued [Cerf, 1974] that hosts should perform this function rather than the communications subnet, but it is not clear that many hosts would be able to support it. However, the protocol does allow the receiver to ignore messages that arrive out of order and rely on the retransmission of the messages to insure that they eventually arrive in order. There are serious questions as to what this would do to the network bandwidth.

As yet only one experiment has been performed using an early version of the Inter-Net protocol [Mader, 1974]. This experiment did uncover a race condition under certain conditions related to establishing a connection; a solution was found to this problem. Otherwise, no major difficulties have been encountered and it is still too early for evaluation of the protocol. The other two major protocols that should be mentioned are also SPAR protocols. The transport protocol of the CYCLADES network [Zimmerman and Elie, 1974] was a precursor to the Inter-Net protocol as was the Virtual Call protocol of Bell Canada's Datapac network [Martel et al., 1974]. The interested reader should be aware of these protocols and how they contributed to the Inter-Net protocol.

Telecommunications Network Protocols (Telnet). One of the earliest protocols that is usually developed using the facilities of the host-host protocol is a Telnet. (The term Telnet is borrowed from the ARPA network jargon. The name has had fairly widespread use and for convenience we will refer to the class of Telnet-like mechanisms by that name, except when distinctions are necessary.) A Telnet protocol provides a computer network with a mechanism by which terminals at remote hosts may gain access to a local time-sharing system much as if they attach to it over local lines.

There are a great variety of terminals, all with different operating conventions. If all hosts on the network had to support all types of terminals, this would cause a great burden on the hosts. This problem has been solved by defining what is known as the Network Virtual Terminal (NVT). The definition of the NVT specifies operating characteristics (such as character set) of this Platonic ideal device. The Telnet protocol requires that all local terminal-like devices must be made to look like a NVT. This means that each network host need only know about the NVT to be able to accept any user terminal. Of course, the NVT form does not encompass the capabilities of all terminals. But it does not exclude them either. All of the Telnet-like protocols described here are based on this NVT concept and, in addition, include a mechanism for what is called option negotiation. This mechanism allows either the user end or the server end of the connection to ask the other side to perform some function that is not specifically covered by the NVT, such as setting tabs, or controlling character echoing. If the other side has never heard of the option, it declines the option; otherwise it accepts the negotiation and proceeds. This mechanism is very flexible and causes no burden on any system.

There are three major Telnet-like protocols of interest: the ARPANET Telnet, the British Experimental Packet Switched Service (EPSS) network's Interactive Terminal Protocol (ITP), and the French CYCLADES network's Virtual Terminal Protocol (VTP). The ARPANET Telnet protocol is the oldest and has been through two major revisions [Postel, 1972], [McKenzie, 1973]. The Telnet NVT is, in essence, a full ASCII terminal with a few extra function keys to abort output, interrupt a process, erase a line, delete a character, etc. (Although these functions exist on most systems, there is little agreement as to what control characters they are.) This definition allows the locally defined function that the user is familiar with to be mapped to a particular character outside the ASCII character set. The Telnet protocol specifies little else. Other

functions are left as options. The only major complaint to be leveled at this protocol is that the data stream format is such that each character in its input stream must be examined in order to find any control characters relevant to the protocol. This very valid complaint has been treated by the CYCLADES VTP.

The French CYCLADES Virtual Terminal Protocol (VTP) [Viver and Zimmerman, 1974] uses the format: <message length> <message> <message length> <message> The first byte after the length indicates whether or not this is a control message. In this way, the necessity of looking at each character is avoided. The VTP introduces another interesting concept. The data stream is considered to be a book made up of pages, which are made up of lines, which are made up of characters. The first byte of a text message indicates whether what is to follow is a new page, a new line, or an addendum to a line. Options are planned for controlling page and line size, and for moving the address pointers. The latter should be good for utilizing terminals with cursor addressing and similiar applications. The number of books and the disposition of the address pointers involved in a VTP connection depend on the characteristics of the devices: for a one way connection, between a process and a line printer, for instance, there is one book with one set of address pointers held by the line printer; for a half duplex connection there is one book with two sets of address pointers, one for each end with a synchronization mechanism for coordination; and for full duplex connection there are two books with two sets of pointers, one for each receiving end.

The British EPSS network's Interactive Terminal Protocol (ITP) [Chandler et al., 1974] has several innovative aspects. The ITP manages its own flow control in a crude way. (Both of the other protocols rely on the host-host protocol's flow control.) The ITP destination can tell the source: 1) to stop sending, 2) to send up to and including first terminating character, 3) to send as much as the sender wants, and 4) to send as much as the sender has buffered regardless of whether it contains a terminating character. The protocol also allows the remote process to tell the local user ITP to transmit everything it has when it receives a class of characters. This protocol also imbeds control and data information in the data stream so that each character must be looked at.

File Oriented Protocols. File oriented protocols are not as well understood as the Telnet protocols. This is probably because there has been no real attempt to standardize the major components of a file system. (The Telnets had a good start due to the existence of standard character sets.)

File protocols have been developed for the three networks that were discussed in the Telnet section. However, we have not been able to find any documentation describing the CYCLADES protocol. Of the other two, the ARPA File Transfer Protocol (FTP) is the more comprehensive. The EPSS FTP does include one facility that the ARPA version of FTP does not. The EPSS FTP [Chandler et al., 1974] is built upon a Data Transfer Protocol (DTP) which oversees the actual transmission of data while the FTP is concerned with problems at a higher level. The advantage of a

DTP is that other protocols that need to move data from one point to another can make use of it, for example, Remote Job Entry (RJE). A DTP was considered at one point in the history of the ARPANET, but was dropped for reasons of expediency. But the idea may be regaining some currency.

Another protocol that has been proposed for the ARPANET is a File Access Protocol (FAP) [Day, 1973]. This protocol considers the problem of addressing and retrieving data within a file (i.e., network random access files). The ARPA FTP is only capable of moving a whole file from one place to another. For many applications, it is prohibitively expensive both in money and time to do this; therefore, the FAP allows selected portions of a file to be retrieved and stored. This protocol has been implemented within the ARPA FTP on at least two sites and seems to work well.

There are four major areas of difficulty with file protocols in a heterogeneous network: 1) file-naming conventions, 2) access-control conventions, 3) the structure of replies to commands, and 4) suitability of the protocol for easy use by both humans and automata. The variety of file naming syntaxes seems at times almost endless. The ARPA Network Working Group (NWG) responded to this problem by defining file name in the Network Virtual File System (NVFS) as a locally recognizable string terminated by a blank. This definition, although clearly a hedge, was expedient and provided a chance to gain experience. Recently, a more sophisticated definition of a general pathname has been suggested [Crocker, 1974]. There are similar difficulties with handling access control in file protocols. So far no protocol has attempted to address the problems of setting or changing access controls on files. The kinds of access control and their scope in different operating systems makes it difficult to characterize access controls in a general, but consistent, way.

Probably the thorniest problem in the ARPA FTP has been that of the replies to the FTP commands. The problem is two-fold: On the one hand, it is mandatory that text be sent back to the user and that the actual contents of the text be determined by the site generating the reply; i.e., the server. This means that in the event of an error the server can send back text explaining the error in terms relevant to the server's file system and intelligible to the user. On the other hand, it is highly desirable for FTP implementations to be capable of being driven by an automaton. Automata can't, of course, be expected to understand English; therefore, a machine readable form must be supplied that does not require the automaton to be too complex. After much discussion, an acceptable reply structure [Postel, 1974 (b)] satisfying most of the necessary requirements has been arrived at by the ARPA FTP group.

Load Sharing Techniques. An obvious potential advantage of a resource sharing network is the possibility of distributing the workload among the systems. However, load sharing in a heterogeneous network environment is not easy. As yet no one has done it, and there are several major obstacles.

One of the obstacles is the difficulty of deciding when to share and, given several possible alternative machines where the program

can be run, which machine to choose. Farber et al. [1973] suggest a scheme for the DCS ring network in which the process wishing to have a function performed for it broadcasts a request for service. The other processors respond with bids for the service which the requestor then surveys to make its choice according to its own criteria. This technique has several advantages, such as distributed control and implicit load balancing. However, as yet no one has suggested what the criteria are for producing a bid or what criteria a requestor might use to evaluate bids. Another obstacle is that there are very few (if any) good indicators of machine load presently in use and none that can be used in a meaningful way across different machines. It is also necessary to be able to predict the load (at least in the short term) on the prospective machines, so that sudden fluctuations in load don't make a good choice become a bad choice in the time it takes to transfer the task.

Once these hurdles have been jumped, there is the problem of how to share tasks. There are essentially three basic techniques. (1) In a homogeneous environment, it may be possible to move the actual machine language files and necessary data to the remote site. This technique is not very practical, in general, and pretty much precludes the ability to make use of specialized systems (e.g., array processors) for certain classes of programs. (2) The source code and data may be moved to the remote host. This is essentially the standard RJE setup. This technique should be workable as long as the remote host has a compiler for the language the program is written in. However, the lack of standardization among compilers can be quite large even on the same machine [Lucas and Presser, 1973]. Also, the different number representations and word lengths on the various computers may introduce errors. This method does incur more overhead since the program must be compiled and run on the remote machine. This consideration must be taken into account in the decision on where the job should be shared. (3) Finally, an equivalent function on the remote machine may be invoked. This approach is more straightforward than the one just mentioned. This technique is being used by the National Software Works' Procedure Call Protocol [White, 1975], [Postel, 1975 (a)]. The main difficulty with this method is validating that the two pieces of code on the different machines do the same thing. This is especially crucial with locally supplied packages, such as statistical packages, where side effects and other subtle differences in implementation can lead to widely differing results for some applications. There is also the problem of converting number representations and of identifying any errors that may be incurred through conversion. In general, any set of calculations that are using the accuracy of the machine near its limits should probably not be shared; however, as long as all calculations and number representations are made in the same way, most errors will cancel.

Remote Job Entry Protocols. Most networks provide this facility if no other. The ARPANET presently has two RJE protocols. One was developed to provide RJE service to the UCLA IBM 360/91 as a matter of expediency [Braden, 1971]; and the other is the official ARPA NETRJE protocol [Bressler et al., 1972]. In what follows we will limit our discussion to the NETRJE protocol. There have been some problems attempting to provide the same sort of service that the traditional RJE

terminal provides. The NETRJE protocol attempts to achieve the best of all possible worlds in a politically untenable environment. The NETRJE attempts to return the user's output to his local site automatically without requiring the user to maintain a connection until the job is complete. This requires the NETRJE Server to store the usercode and password of the user for the site where the output is destined to go. This is an unconscionable breach of the site security. The security problem cannot be averted with the NETRJE structure in a network that does not have a network-wide authentication system. Also, the NETRJE requires both sites to have implemented FTP, which NETRJE uses to move the jobs over the net. This can place a great burden on some systems that do not have the machine resources or manpower to implement both protocols. Most of these problems have been addressed and solved in a proposed new RJE protocol [Day and Grossman, 1975].

Protocol Formalisms. The preceding descriptions of the protocols found in most contemporary networks should provide a glimpse of their complexity. Because of this complexity it is quite often difficult to ascertain the properties and weak points of a protocol without implementing it, and even implementation can be plagued with pitfalls. The designers of a protocol for a heterogeneous computer network are confronted with the additional problem of defining the protocol unambiguously. These problems have led recently to increased interest in formal methods for defining and analyzing communication protocols.

Some of the earliest work was done by Petri [1962], who was interested in formally describing systems of communicating automata. He introduced to traditional automata theory several constraints that more closely reflect the real world. For example, Petri assumes that there is an upper bound on the propagation speed of signals and on the density with which information can be stored. In addition, he notes that fixed, finite automata can recognize at best iteratively defined classes of functions; those recursive functions that can't be made iterative require automata of unbounded size. He then uses this theoretical framework to prove many properties of asynchronously cooperating automata. Probably the best known result from this work is the so-called "Petri Net", which is used for analyzing dependencies of asynchronously cooperating processes in operating systems.

Postel [1974 (a)] deals more directly with the problems related to protocol design. Postel uses what is called the "UCLA graph model" to analyze ARPANET-like host-host protocols. This technique can lead to very complex graphs if applied directly, but Postel shows that the use of "graph modules" can simplify the graphs considerably. The methodology developed can be of utility in the design and checking of the flow of control in a protocol for deadlocks or critical race conditions.

An approach similar to the two noted above is being taken by a group at the University of Wisconsin [Fitzwater and Smith, 1973]. They have developed a language for describing asynchronous complexes of interacting digital systems at virtually any level of abstraction. Their technique is based on the Post production systems and uses a string representation as a happy medium between graph models and integer

representations (Godel numbers). One of the major results is the development of an iterative, top-down technique for analyzing such systems so that a trade-off may be made between the cost of the analysis and the detail of the analysis [Smith and Fitzwater, 1974].

The above techniques can represent potential deadlocks and race conditions that would occur in an error free system, but can not be used to discover potential deadlocks or races in an error-prone system where messages may become garbled, lost, or duplicated. The Stanford Digital Systems group has just begun investigating techniques for analyzing this condition [Sunshine, 1974]. Early work appears to be heavily oriented toward proving theorems about a small class of protocols known as Positive ACK/Retransmission (PAR) protocols and Sequencing PAR (SPAR) protocols. It is much too early to be able to say what the outcome of this work will be.

As yet no one has really attacked the problem of a method of specifying a protocol, especially the higher level ones, in a way that is suitable as a base for an implementation. Although the above methods are applicable, the definitions derived are sufficiently complex for any reasonable protocol as to make them impractical by themselves.

Distributed Operating Systems. The previous discussion of the problems of establishing connections and manipulating terminals, files, and job streams leads to the obvious generalization that the proper paradigm for organizing the functions required to support network applications is a distributed operating system. Such a paradigm would provide a unified approach to supporting the operating system functions normally required by user applications in such a way that extensive re-coding is not necessary.

Metcalfe [1973] briefly discusses one basis for such a system which is termed the best-effort thin-line approach. Metcalfe contrasts "thin-line" and "thick-line" forms of interprocess communication. By "thick-line" he is referring to those methods of interprocess communication found in most operating systems. These are usually characterized by a centralized organization and a very high bandwidth for accessing local resources. In the "thin-line" approach, a process communicates with other processes and accesses resources as if they were remote. This approach does tend toward more modular systems, which has a clarifying effect on the management of multi-process activity and extends easily as systems become more distributed. The term "best effort" implies that a part of a larger system attempts to continue to perform its function as well as it can in the face of failures elsewhere in the system. The "thin-line" approach can lead to a highly modular and reliable system that degrades gracefully when parts of it fail. As yet there is little experience with systems constructed on these criteria and it is not really clear how some components of a system should be represented in such a scheme.

There are presently two research efforts into what might be called distributed operating systems. One is the RSEXEC (Resource Sharing Executive) system [Thomas, 1973] and the other is the National Software Works [Balzer et al., 1973]. RSEXEC has been an ongoing

effort at BBN for about three years. The system was initially used among the TENEX's on the ARPA network, but was later modified to be suitable for other machines. RSEXEC presently provides a user with the ability to find the location of people on the net, to open communication with them (either directly or by mail) , and to set up teleconferencing sessions. Recent work has centered on providing a distributed file system and an environment for executing programs [Thomas, 1974].

The purpose of the National Software Works (NSW) is to provide a facility for developing production software for conventional ADP (i.e. COBOL) applications. This system consists of essentially four parts: the ARPANET, the datacomputer (a trillion bit store and associated retrieval system), the execution machine, and the front-end. The term front-end means something a little different here than in most uses of the phrase. Here it refers to a front-end to the NSW system, not to a particular machine. The front-end for the NSW provides support for a powerful user interface with large high-speed screen displays and a command language that may be molded to suit the user's situation. One of the goals of the NSW is to provide a user with sophisticated tools for editing, proving, and debugging programs. These tools will exist on various hosts on the network, and the user should be totally unaware that he is using the network [Balzer et al., 1973]. The basis for the network interaction is the Procedure Call Protocol (PCP) [White, 1975]. This protocol allows the user's local program to "call" procedures that exist on other systems and to pass data to them. This system then serves as a basis for the NSW protocols [Postel, 1975 (a)] that provide the distributed NSW tool environment.

The work on distributed operating systems is just beginning and as yet it is too early for any conclusions to be made. None of the systems mentioned are sufficiently far in their development to have run into the really nasty problems related to synchronization and updating of a file in a failure-prone system. The latter problem has just been tackled by the RSEXEC effort.

Summary and Assessment

Many of the problems that have arisen in computer networking stem from the problems involved in maintaining duplicate copies of information in a distributed environment. Many of these problems have always been around, but some aspect (such as delay) of computer networks has made them crucial. Little work has been done on the general problem, and for a while, at least, specific solutions may be necessary in practice. This is an area which badly needs unification and further research.

Deadlock is a fairly well understood subject in computer systems, though the implications for distributed environments have only recently begun to become clear. Uniform naming schemes for processes and resources in a network, and suitable cooperation between the systems involved can give acceptable levels of protection from the problems of deadlock in a distributed network of computers. However, such naming schemes and cooperation have only begun to appear.

Resource naming is an apparently esoteric topic which almost always appears well-disguised when encountered in practice. The ARPANET

Initial Connection Protocol (ICP) contains several examples of methods that are used in practice to manipulate name spaces, as does the ARPANET host-host protocol (cf. Protocol section). Deadlock prevention and detection schemes for networks ([Somia, 1973], [Chu and Ohlmacher, 1974]) often make implicit assumptions about names. These assumptions include either uniform process naming or diverse methods for referring to processes, depending on the computer system involved. Directories and catalogs, which only serve to define name spaces and information about them, are normally treated as completely different topics from virtual memory schemes, for example, which do the same thing. Some generalization of naming and name space manipulation techniques might permit methods to be developed which could handle several problems currently treated separately.

Directories and catalogs in distributed environments are a topic of some concern at present. Reliability considerations dictate that multiple copies be kept, but the present technology has trouble supporting this seemingly simple requirement. The need for specific solutions which are efficient, inexpensive, and reliable, will continue to increase.

There are still many unanswered questions with respect to the design and implementation of host-host level protocols. One of the largest barriers to gaining insight into these problems, as Postel [1974 (c)] found out, is that very few protocol implementations are instrumented at all. Thus, it is difficult to ascertain what causes the observed protocol behavior. (It is heartening to note that the definition of Inter-Net Protocol included measurements [Cerf et al., 1974].) Further investigation is needed into how these protocols can be made resilient and less susceptible to error conditions. Also a better understanding of flow control strategies is needed with special emphasis on how the load conditions of the host should affect such strategies. There has also been some discussion of the host's doing packet reassembly and thus avoiding the reassembly lockup problem. (See the Communication Network Design Issues section of this report or [Cerf, 1974].) The advantages and disadvantages are not at all well understood and should be considered in greater detail.

Telnet protocol development is fairly well understood and the existing designs fulfill the necessary basic requirements. The option negotiation mechanism, which is found in some form in each of the Telnet protocols discussed above, is a clean and elegant way of making the protocol extensible and flexible. There is some question as to whether the negotiation mechanism belongs in Telnet or should be considered in a much broader context. The implications of this question have yet to be pursued.

File protocols still need quite a bit of research. The concept of the Virtual File System has been slow to develop because of the problems mentioned above. The forms of access control and the scope of the attributes for various systems need to be studied closely to determine how access control functions can be or should be included in file protocols. The utility of a Data Transfer Protocol in the context of

more sophisticated protocol applications needs to be considered in some detail.

Load sharing techniques are by far the least well understood with respect to the remote accessing of resources. Most of the problems in load sharing will be clarified as a better understanding of how to measure system load is developed, and as careful investigations of the incompatibilities of number representations, compilers, and other user-library functions are made. The formal analysis of network protocols is in an early state of development. Problems related to the flow of control (such as race conditions) can be analyzed, although the techniques are not in general use. Other problems dependent on errors in the system or traffic conditions are not well understood, and as yet no formal methods really exist for analyzing these conditions.

The concept of a distributed operating system has been kicking around in bull sessions for some time, but actual work on it has only begun in earnest in the last year. A design requirements is to develop a sufficiently general model for the distributed operating system. The model should encompass the important attributes of the operating systems comprising the network. There has been very little work on this concept in the global sense, but some of the canonical forms developed for other protocols (e.g., the Network Virtual Terminal) can be applied or used as the groundwork for more general characterizations.

References

- Balzer, R, Cheatham, T.E., Crocker, S., and Warshall, S.
1973 "Design of a National Software Works", USC/Information Sciences Institute ISI-RR-73-16.
- Bouknight, W.J., Grossman, G.R., and Grothe, D.M.
1973 "The ARPA Network Terminal System, A New Approach to Network Access" in Data Networks, Analysis and Design, Proceedings of the 3rd Data Communications Symp., pp. 73-79.
- Braden, R.
1971 "Interim NETRJS Specification", RFC #189.
- Bressler, R., Murphy, D., and Walden, D.
1972 "A Proposed Experiment with Message Switch Protocols" RFC #333.
- Bressler, R., Guida, R., and McKenzie, A.A.
1972 "Remote Job Entry Protocol" RFC #360.
- Burchfiel, J., and Tomilson, R.
1973 "Proposed Change to Host-Host Protocol - Resynchronization of Connection Status", RFC 467.
- Carr, S., and Crocker, S.
1970 "Host-Host Communication Protocol in the ARPA Networks", AFIPS 39.
- Cerf, V., Yogen, D., and Sunshine, C.
1974 "Specification of Internet Transmission Control Program" INWG Note #72.
- Cerf, V.
1974 "Assessments of ARPANET Protocols", RFC #635.
- Chandler, A.S., Adams, C.J., Barry, P.T., Dewis, I.G. and Hammond, N.R., Higginson, P.L., Johb, R.B., and Olejniczak, J.J.
1974 "Report of the Higher Level Protocol Working Group", INWG Note #6.
- Chu, W.W.
1975 Personal Communication, University of California at Los Angeles, Los Angeles, Ca.
- Chu, W.W., and Ohlmacher, G.
1974 "Avoiding Deadlock in Distributed Data Bases", Proceedings of the ACM National Symposium 1, pp. 156-160.
- Crocker, D.
1974 "Network Standard Data Specification Syntax", RFC 645.
- Day, J.
1973 "Proposal for File Access Protocol", RFC #520.

Day, J., and Grossman, G.R.

1975 "A Proposed Remote Job Entry Protocol", RFC in preparation.

Dijkstra, E.W.

1968 "Co-operating Sequential Processes", in Programming Languages, F. Genuy ed., Academic Press, N.Y., 1968.

Farber, D.J., Feldman, J., Heinrich, F.R., Hopwood, M.D., Larson, D.C., Loomis, D.C., and Rowe, L.A.

1973 "The Distributed Computing System", Proc. Seventh Annual IEEE Computer Society International Conf.

Farber, D.J., Heinrich, F.R.

1972 "The Structure of the Distributed Computer System - The Distributed File System", Network Problems II, Proceedings of the ICCS.

Farber, D.J., and Larson, K.

1972 (a) "The Structure of a Distributed Computer System - Communications", Proceedings of the Symposium on Computer-Communications Networks and Teletraffic, J. Fox, ed., Microwave Research Institute of Polytechnic Institute of Brooklyn, pp. 21-27.

(b) "The Structure of a Distributed Computer System - Software", Proceedings of the Symposium on Computer-Communications Network and Teletraffic, J. Fox, ed., Microwave Research Institute of Polytechnic Institute of Brooklyn, pp. 539-545.

Fitzwater, D.W., and Smith, P.Z.

1973 "A Formal Definition Universe for Complexes of Interacting Digital Systems", Univ. of Wisconsin - Madison, Computer Sciences Technical Report #184.

Hathaway, W.

1973 "More on Lost Message Detection", RFC 512.

Havender, J.W.

1968 "Avoiding Deadlock in Multitasking Systems", IBM Syst. Journ. 7, No. 2, pp. 74-84 (Indirect Reference).

Holt, R.C.

1971 "Comments on Prevention of System Deadlocks", (ACM 14, pp. 36-38.

Johnson, P.R., Thomas, R.H.

1975 "The Maintenance of Duplicate Databases", RFC #677, NIC #31507, Jan. 1975.

Kanodia, R.K.

1974 "A Lost Message Detection and Recovery Protocol", RFC #663.

- Knott, G.D.
1974 "A Proposal for Certain Process Management and Intercommunication Primitives", Operating Systems Review 8, Sec. 1-6, Oct. 1974, ACM Special Interest Group on Operating Systems, pp. 7-44.
- Knuth, D.E.
1966 "Additional Comments on a Problem in Concurrent Programming Control", (ACM 9, pp. 321-322, (Indirect reference).
- Kosaraju, S.R.
1973 "Limitations of Dijkstra's Semaphore Primitives and Petri Nets", Operating System Reviews 7, Oct. 1973, ACM Special Interest Group on Operating Systems, pp. 122-126.
- Lamport, L.
1974 "A New Solution of Dijkstra's Concurrent Programming Problem" CACM 17, pp. 453-455.
- Lucas, H.C., and Presser, L.
1973 "A Method of Software Evaluation: The Case of Programming Language Translators", Computer Journal 16, pp. 226-231.
- Mader, E.
1974 "A Protocol Experiment", RFC #700.
- Martel, C.C., Cunningham, I.M., and Grushcow, M.S.
1974 "The BNR Network: A Canadian Experience with Packet Switching Technology", Proc. IFIP.
- McKenzie, A.A.
1972 "Host/Host Protocol for the ARPA Network", NIC #8246.
- McKenzie, A.A.
1973 "Telnet Protocol Specification", NIC #15372.
- Metcalfe, R.
1973 "Packet Communication", Project MAC, Technical Report. TR-114, Massachusetts Institute of Technology.
- Meyer, E.
1973 "Response to RFC 467", RFC #493.
- Miller, R.E.
1974 "Some Relationships Between Various Models of Parallelism and Synchronization", IBM Research Document RC 5074 (#22391), IBM Corp., Yorktown Heights, N.Y.
- Miller, T.J.
1974 "Deadlock in Distributed Computer Networks", M.S. Thesis, Department of Computer Science Report UIUCDCS-R-74-619, University of Illinois at Urbana-Champaign, Urbana, Il.
- Neigus, N.
1973 "File Transfer Protocol", RFC #542.

Patil, S.S.

- 1971 "Limitations and Capabilities of Dijkstra's Semaphore Primitives for Coordination Among Processes", Project MAC Memo #57, Massachusetts Institute of Technology, Cambridge, Mass.

Petri, C.A.

- 1962 "Communication with Automata", Thesis, Darmstadt Institute of Technology, Bonn, Germany.

Postel, J. B.

- 1972 "Telnet Protocol Specification", RFC #318.

Postel, J.

- 1974 (a) "A Graph Model Analysis of Computer Communications Protocols", Thesis, UCLA.

(b) "Revised FTP Reply Codes", RFC #640.

(c) "Survey of Network Control Programs in the ARPA Computer Network", Mitre Technical Report 6722, Rev. 1, Mitre Corporation, McLean, Va.

- 1975 (a) "National Software Works Protocols", Augmentation Research Center, Stanford Research Institute, Menlo Park, Ca.

(b) Personal Communication, Jan. 1975, Augmentation Research Center, Stanford Research Institute, Menlo Park, Ca.

Pouzin, L.

- 1973 "Presentation and Major Design Aspects of the CYCLADES Computer Network", 3rd Data Communications Symposium.

Retz, D.L., Miller, J.R., McClurg, J.L., and Schafer, B.W.

- 1974 "ELF System Programmer's Guide", Speech Communications Research Lab, Santa Barbara, Ca.

Roberts, L.G., Wessler, B.D.

- 1970 "Computer Network Development to Achieve Resource Sharing", Proceedings of the SJCC, IFIPS, pp. 543-549.

Saltzer, J.H.

- 1974 Personal Communication with Peter A. Alsberg.

Shaw, A.C.

- 1974 The Logical Design of Operating Systems, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Smith, P.Z., and Fitzwater, D.R.

- 1974 "Efficient Analysis of the Process Structures of Formally Defined Complexes of Interacting Digital Systems", University of Wisconsin-Madison, Computer Sciences Technical Report #219.

- Somia, M.
1973 "Synchronization Problems in a Computer Network", Int'l. Computer Symposium, A. Gunther et al., eds., North Holland Pub. Co.
- Sunshine, C.
1974 "Issues in Communication Protocol Design - - Formal Correctness", INWG Protocol Note #5.
- Thomas, R.
1973 "A Resource Sharing Executive for the ARPANET", BBN Report 2522, Bolt, Beranek, and Newman, Cambridge, Mass.
- Thomas, R.
1974 Personal Communication
- van Lamsweerde, A.
1974 "Deadlock Prevention in Real Time Systems", International Computing Symposium 1973, A. Gunther, et al., eds., North Holland Publishing Co.
- Viver, A. and Zimmerman, H.
1974 "Virtual Terminal Protocol (VTP) Proposed Specifications", Reseau CYCLADES, TER 503.1.
- Walden, D.
1972 "A System for Interprocess Communication in a Resource Sharing Computer Network". CACM 15, pp. 221-230.
1973 "Lost Message Detection", RFC #534.
- White, J.E.
1975 "The Procedure Call Protocol - Version 2", NIC Document #24459.
- Wodon, P.L.
1972 "Still Another Tool for Synchronizing Cooperating Processes", Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa.
- Zimmerman, H., and Elie, M.
1974 "Transport Protocol Standard Host-Host Protocol for Heterogeneous Computer Networks", Reseau CYCLADES, SCH 519.1

Introduction

There are two broad aspects to the measurement and evaluation of computer systems. The first, which is also the most highly developed, is performance evaluation to aid in the decision making of system designers and computer users. The system designer requires detailed measurements to indicate to him where redesign or tuning is needed. On the other hand, the system administrator or user is interested in coarser measures of system performance such as throughput or response time. This aspect can itself be considered as having two parts: analytical techniques represented by statistical and queuing theoretic methods and the black box techniques represented by benchmarks or synthetic jobs. The second major aspect of performance measurement is concerned with determining or predicting machine load in the short term for purposes of job scheduling and utilizing system resources efficiently.

The development of performance measurement and evaluation techniques has been slow. In fact, it has not been until the last three to five years that there has been any major interest in the field. Up to that time, a few isolated papers appeared from time to time, but they were usually highly specific to the system or environment for which the measurements were done. This wouldn't have been so bad if the same or similar measurements had been done for other systems, so that a corpus of data would have been built up from which generalizations could be made. Quite often, though, measurements were left as isolated observations and the questions raised were never followed up. The reasons for this longstanding state of affairs are many, but can be roughly characterized by the following three observations. First, during this early period there was much more emphasis placed on making the systems fast and small than on finding out why they weren't. There was very little willingness to spend resources on measurement. Second, the formal mathematics to model systems did not exist. And third, very few people in the field knew anything about the existing mathematics that was applicable or the difficulties of setting up an experiment. In fact, this is still a major stumbling block.

The problem is not, as many would lead one to believe, that the systems are so overwhelmingly complex, but that there are so many things that can be measured that it was (and still is) difficult, given the present state of knowledge, to know what should be measured. Deciding what measurements to take and where to take them is a very subtle problem; and once the measurements are taken, interpreting them can require even more subtlety.

The last three-to-five years has seen an increased interest in the field. This does not imply that there have been major breakthroughs, but a moderate amount of progress has been made. The mathematical theories that are being developed have reached sufficient power and sophistication so that more accurate models may be constructed. Also, the generally higher level of sophistication of computer users, who are now demanding performance evaluation of the systems they use or intend to use, has caused a stronger interest in measurement by both vendors and users. (This trend has a long way to go.)

Performance Evaluation

Most performance evaluation techniques fall into one of two basic categories. Either the system is treated as a black box and measures are taken that do not attempt to understand the internal workings of the system, or the internal structure is modelled in an attempt to understand how the system's behavior arises. This latter approach has been termed analytical by Karush [1969]. Under the analytical approach, several general methods may be cited: queuing theoretic models, probabilistic models, and simulations. The black box approach is represented by benchmarks and synthetic job streams. Each of these will be discussed below.

Analytical Approach. Much of the work has centered around queuing models of systems. An operating system can be characterized as an open network of queues for purposes of analyzing response time, throughput, etc. However, it has only been recently that the mathematics was sufficiently developed to allow analysis of the network models. Initially, single-queue or single-queue-with-feedback systems were analyzed as models of processor utilization. These systems were analyzed for several queuing disciplines, such as first-come-first-served, round-robin, foreground-background, and priority scheduling, to yield basic results on response time and throughput [Kleinrock, 1975]. For the queuing models, the inter-arrival times are generally considered to be Poisson distributed and service times to be exponentially distributed. (Cf. discussion of queuing theory (p. 73) in this report.)

The single-queue models were, however, inadequate for describing computer systems, in which processes move around in a network of input/output queues, central processor queues, etc. Some of the earliest applicable mathematical work on queuing networks was done by Gordon and Newell [1967 (b)]. Based on this theoretical work, models for computer operating systems were first devised by Kimbleton [1971] and Arora and Gallo [1971]. Generally a cyclic queuing model of an operating system includes the queues of the running processes, the blocked processes (waiting for an I/O) and the ready processes [Kimbleton, 1971] (Arora uses a similar breakdown, but also attempts to model memory hierarchies). This allows some insight into the internal workings of the system. (The single queue model has an essentially black box viewpoint.) A cyclic queue model, however, tends to lump several queues together. This can lead to disagreement between predictions from the model and the observed system behavior [Kimbleton, 1971].

Some work has been done on more general networks of queues based on the work of Gordon and Newell [1967 (a)]. Buzen [1971] has worked out such a model of a computer system and looked at the problem of relating queue lengths to system throughput (again for the exponential case only). Pewitt and Su [1973] have attempted to use this model as a basis for system scheduling, but have found it somewhat difficult to extract the needed information. Kobayashi has taken a unique approach and applied diffusion approximation techniques to this kind of queuing network with significant results for response time, throughput and queue length probabilities [Kobayashi, 1973 (a), (b)]. The diffusion approximation also gives more accurate results when the queues are described by non-exponential servers than do the normal queuing models [Reiser and Kobayashi, 1973].

When the mathematics necessary to solve some of these problems becomes intractable, or simply does not exist, it is often necessary to resort to simulations. Simulations present their own set of problems, however. The

biggest of these is showing that a particular simulation does indeed accurately model the system for the desired measurements. (It is much more difficult, it seems, to notice the points where the simulation differs from reality than to notice such points with a mathematical model. This is probably because simulations attempt to model many more features than a mathematical model will attempt.) Much work on simulations has been done, most of it highly specific to a particular problem. However, the work of Nielsen [1967] is a good example of a more general approach to simulation. Nielsen describes a general model capable of simulating most time-sharing systems. This is then used to model the IBM 360/67. Nielsen is able to show where many system bottlenecks occur and what actions will alleviate them.

In some cases a queuing model does not allow one to see the particular relations of interest; more general probabilistic techniques are usually then applied. An example is the modeling of an interactive user population. Estrin and Kleinrock [1967] were able to use a queuing model to get results on the response time to Poisson-distributed user requests to an operating system; however, this does not give any insight into the demand on system resources. To do this Denning [1968 (a)] used a statistical analysis to get some very good results on such things as number of processes waiting on input, maximum process execution rate, and bounds on buffer requirements.

Black Box Approach. The black box approach solves the problems of model accuracy by using representative programs to generate the characteristic behavior of the application environment; the actual system is the "simulator".

The technique of using benchmarks or synthetic job streams is a well-established but poorly understood approach to system measurement. The benchmark has long been the trustworthy tool of the system administrator who is trying to compare one machine with another. It has the distinct advantages that intimate knowledge of the system is not necessary for the performance of independent measures and that the machine can be tested for the desired class of computing tasks. The benchmark has been discussed to a large degree in the literature [Karush, 1969], [Buchholz, 1969], [Kernighan and Hamilton, 1973], and [Morgan, 1973]. However, most benchmark work has been highly specific to either the machine or the user's application environment. The synthetic job holds more promise of having general applicability [Buchholz, 1969]. A synthetic job is a program which has the same running characteristics (pattern of CPU time, paging, and I/O operations) as the desired job stream except that the program does no useful work. The point is that it should be much easier and less expensive to write a program for a machine to act in a particular way, than it would be to transfer a large software system that does useful work to produce the same system load to the machine being tested. Generally, such programs are parameterized so that they can be used to simulate a large variety of program types. This kind of approach shows much promise. However, the design of a synthetic job can be very tricky. For example, the parameterized synthetic job described by Buchholz [1969] does not measure procedure-call overhead or array indexing efficiency. This latter point could cause unreliable measurements when one compares a traditional machine, like an IBM 360, with a descriptor-oriented machine, like a B5500 or Honeywell 6180.

Load Determination and Prediction

Ask any electrical engineer to give you the steady state solution of some circuit and he will whip out his book of Laplace transforms and a few

minutes later give you a nice answer; but ask him what the transient behavior is and he frowns and gnashes his teeth - but does come up with a solution in time. Determining and predicting the load on a system is very analogous to solving for the transient behavior. As one can see from the preceding paragraphs, our understanding of the "steady-state" is not very good, and our understanding of what affects load in the short-term is even worse. Several kinds of transient effects are used as inputs to system tuning. Probably the best known of these are the algorithms for page replacement in paging systems. For paging systems some of the best known replacement algorithms are the so-called least-recently-used (LRU), [Coffman and Varian, 1968], page fault frequency [Chu and Opderbeck, 1972] and the working set [Denning, 1968 (b)]. However, these are fairly restricted measures and can't really be used alone for general system measurement.

Wilkes, Baur and McCredie, and Wulf have all attempted to find measures that were useful to dynamic load control. They use methods borrowed from control theory. Wilkes [1971] describes an automatic load control system for the Compatible Time Sharing System (CTSS) based on an exponentially-decaying average queue length and a measure of swapping overflow as an indication of load. This is then used to compute the number of tasks that should be allowed in the system. Baur and McCredie [1973] applied a control system scheme similar to that of Wilkes, but they use a tandem queuing model based on measures of queue lengths and service times for the CPU and I/O systems. The control algorithm is then used to predict what would happen if another user or job were added to the system. The primary difficulty with these control system techniques is the stability of the control function. Stability is very difficult to prove; and it is not easy to impose boundary conditions on the domain of the function to make the proofs more tractable without introducing unrealistic constraints into the model. In fact, both Wilkes and Baur and McCredie comment that instabilities are indeed possible in practical situations. Wulf [1969] suggests several measures of dynamic load based on processor and channel time per process and the time a process is idle waiting for processor or channel. In addition, proper "estimators" can be applied to these measures for purposes of smoothing them and limiting their "memory". Simplified versions of some of these measures were used on the B5500 for load control and produced a 20-25% increase in processor utilization and virtually eliminated thrashing situations.

Summary and Assessment

The measurement and evaluation of computer system performance is a fairly young field that is just beginning to make its weight felt. Much more work is needed in most of the areas outlined above in order that accurate predictions of computer performance can be made.

The state of development of the queuing models is still fairly primitive. The queuing models represent fairly abstract models of operating systems; it is sometimes difficult to go from theory to practice and vice versa. For example, it is not easy to characterize the various queuing disciplines or resource allocation algorithms in terms of queuing models in such a way as to preserve their more interesting properties. Also, to date no extensive empirical data has been collected to characterize the resource demands of different user environments (such as scientific or business) so that

this information could give some insight into the properties of various designs for particular applications.

The basic difficulty with the present state of analytical measuring and modeling is that the models tend to be so abstract that they are difficult to apply. It is also difficult to draw on the published measurement results for other systems, because these results are often tinted by hidden assumptions about the operating system organization. The fact that these assumptions are not mentioned is not deliberate, but primarily because most computer scientists at present are not familiar with enough different operating system organizations to be aware of how they influence the data.

Similarly, the benchmark is too specific to be able to supply data for the comparison of operating systems. The synthetic program approach, on the other hand, does provide some possibilities. A carefully designed standard synthetic job defined in a high level language could provide a straightforward technique for measuring systems at a low cost, but this problem needs considerably more study.

Load determination and prediction present very difficult problems, primarily with finding good measures and stable control functions. In the context of a network, the problems are compounded. For network load-sharing applications, one would like measures of system load that are machine independent so that a user can find a lightly loaded machine to execute his job. To date very little work has been attempted on this problem.

Probably one of the biggest problems within this field is the lack of a body of data on either machines or programs. Very little work has been done to determine what the running characteristics of classes of programs are. Lore and common sense are usually referred to for guidelines. It is commonly said that "scientific computations are processor bound" and "business computations are I/O bound"; but no one has really obtained measurements on program classes that would be valid as input to an operating system model. The lack of good data can impair understanding at least as much as the lack of a good theoretical framework.

References

- Aho, A.V., Denning, P.J. and Ullman, J.D.
1971 "Principles of Optimal Page Replacement", J. ACM 18, pp. 80-93.
- Arora, S.R. and Gallo, A.
1971 "The Optimal Organization of Multiprogrammed Multi-Level Memory", Proc. ACM Workshop on System Performance Evaluation, pp. 104-141.
- Bauer, M.J. and McCredie, J.W.
1973 "AMS: A Software Monitor, for Performance Evaluation and System Control", Proc. 1st Annual SIGME Symp., pp. 147-160.
- Bruo, J., Coffman, E.G., Jr. and Sethi, R.
1974 "Scheduling Independent Tasks to Reduce Mean Finishing Time", CACM 17, pp. 382-387.
- Buchholz, W.
1969 "A Synthetic Job for Measuring System Performance", IBM System Journal 8, pp. 309-318.
- Burke, P.J.
1956 "The Output of a Queuing System", Op. Res. 4, pp. 699-704.
- Buzen, J.
1971 "Analysis of Systems Bottlenecks Using a Queuing Network Model", Proc. ACM Workshop on System Performance Evaluation, pp. 82-103.
- Cady, G.M.
1972 "Computation and Communication Trade-Off Studies: An Analytical Model of Computer Networks", Proc. WESCON Conference, pp. 1-12.
- Campbell, J.W. and Heffner, W.J.
1968 "Measurement and Analysis of Large Operating Systems During System Development", Proc. FJCC, pp. 903-914.
- Chu, W.W. and Opderbeck, H.
1972 "The Page Fault Frequency Replacement Algorithm", Proc. AFIPS, pp. 597-609.
- Coffman, E.G. and Varian, L.C.
1968 "Further Experimental Data on the Behavior of Programs in a Paging Environment", CACM 11, pp. 471-474.
- Denning, P.J.
1968 (a) "A Statistical Model for Console Behavior in Multiuser Computers", CACM 11, pp. 605-612.
(b) "The Working Set Model for Program Behavior", CACM 11, pp. 323-333
1970 "Virtual Memory", Computing Surveys 2, pp. 153-189.

- Denning, P.J. and Eisenstein, B.A.
 1971 "Statistical Methods in Performance Evaluation", Proc. ACM Workshop on Systems Performance Evaluation, pp. 284-307.
- Estrin, G. and Kleinrock, L.
 1967 "Measures, Models and Measurements for Time-Shared Computer Utilities", Proc. ACM Conf., pp. 85-96.
- Freibergs, I.F.
 1968 "The Dynamic Behavior of Programs", Proc. FJCC, pp. 1163-1167.
- Gelenbe, E., Tiberio, P. and Boekhorst, J.C.A.
 1973 "Page Size in Demand-Paging Systems", Proc. 1st Annual SIGME Symp., pp. 1-12.
- Gordon, W.J. and Newell, G.F.
 1967 (a) "Closed Queuing Systems with Exponential Servers", Op. Res. 15, pp. 254-265.
 (b) "Cyclic Queuing Systems with Restricted Queue Lengths", Op. Res. 15, pp. 266-276.
- Jackson, J.R.
 1957 "Networks of Waiting Lines", Op. Res. 3, pp. 518-521.
- Karush, A.D.
 1969 "Two Approaches for Measuring the Performance of Time Sharing Systems", Proc. 2nd Symp. on Operating System Principles, Princeton, pp. 159-166.
- Kernighan, B.W. and Hamilton, P.A.
 1973 "Synthetically Generated Performance Test Loads for Operating Systems", Proc. 1st Annual SIGME Symp., pp. 121-126.
- Kimbleton, S.R. and Moore, C.G.
 1971 "A Probabilistic Framework for System Performance Evaluation", Proc. ACM Workshop on System Performance Evaluation, pp. 337-362.
- Kleinrock, L.
 1970 "A Continuum of Time-Sharing Scheduling Algorithms", Proc. AFIPS SJCC, pp. 453-458.
 1975 Queuing Systems. Vol. 1: Theory, John Wiley, New York.
- Kobayashi, H.
 1973 (a) "Application of the Diffusion Approximation to Queuing Networks Part 1 - Equilibrium Queue Distributions", Proc. 1st Annual SIGME Symp., ACM, pp. 54-63.
 (b) "Application of the Diffusion Approximation to Queueing Networks II: Nonequilibrium Distributions and Applications to Computer Modeling", Journal ACM 21, pp. 459-469.
- Morgan, D.E. and Campbell, J.A.
 1973 "An Answer to a User's Plea?", Proc. 1st Annual SIGME Symp., pp. 112-121.

Nielsen, N.

1967 "The Simulation of Time Sharing Systems", CACM 10, pp. 397-413.

Opderbeck, H. and Chu, W.W.

1974 "Performance of the Page Fault Frequency Replacement Algorithm in a Multiprogramming Environment", Proc. IFIP, North Holland, Amsterdam, pp. 235-241.

Pewitt, T.C. and Su, S.Y.W.

1973 "Resource Demand Paging and Dispatching to Optimize Resource Utilization in an Operating System", Proc. 1st Annual SIGME Symp., pp. 29-36.

Ramamoorthy, C.V., Chandy, K.M. and Gonzalez, M.J.

1972 "Optimal Scheduling Strategies in a Multiprocessor System", IEEE Transactions on Computers C-21, pp. 137-146.

Reiser, M. and Kobayashi, H.

1974 "Accuracy of the Diffusion Approximation for Some Queuing Systems", IBM Journal of Research and Development 18, pp. 110-124.

Rozawadowski, R.T.

1973 "A Measure for the Quantity of Computation", Proc. 1st Annual SIGME Symp., pp. 100-111.

Sekino, A.

1973 "Throughput Analysis of Multiprogrammed Virtual Memory Computer Systems", Proc. 1st Annual SIGME Symp., pp. 47-53.

Thurber, K.J. and Jack, L.A.

1973 "Time Driven Scheduling", Proc. IEEE Comp. Conf., pp. 181-184.

Waldbaum, G.

1973 "Evaluating Computing System Changes by Means of Regression Models", Proc. 1st Annual SIGME Symp., pp. 127-135.

Wilkes, M.V.

1971 "Automatic Load Leveling in Time Sharing Systems", Proc. ACM Workshop on System Performance Evaluation, pp. 308-321.

Wulf, W.A.

1969 "Performance Monitors for Multi-Programming Systems", Proc. 2nd Symp. on Operating Systems Principles, Princeton, pp. 175-181.

Introduction

The last few years in the field of computer networking have seen a growing interest in two related developments: the front-end and the network access system. The front-end normally consists of a mini-computer (and possibly a few peripherals such as terminals or tapes) connected both to the network and (by a special high-speed interface) to the larger general purpose system or host. (In what follows the machine being front-ended will be referred to as the host.) The interest in the front-end grew from primarily economic considerations. Experience showed that there are several difficulties with attaching some computers to ARPA-like networks. Some machine architectures do not lend themselves to the communication and real time constraints required by the network software. In other cases the network software requires too many system resources. Implementation of necessary network software may not be feasible due to the lack of manpower or expertise or political considerations (e.g., implementation of network software requires modification of vendor software). For prospective hosts in these or similar situations, the front-end provides a viable economic alternative. The necessary network software is implemented in the front-end, along with the implementation of a simpler protocol to the host. Other software to support local devices may also be implemented in the front-end, depending on the application. The required software in the host may be kept small or perhaps may be non-existent for very limited applications. The front-end can therefore be used to remove from the host much of the load incurred by networking. It is also possible in some situations to move much of the host's terminal load to the front-end for users that are primarily using the network.

The network-access system addresses the complementary problem. Here a group wishes to have access to a network but does not have or want a large local machine. A scant three years ago the thought of using computers and doing computer science research without an in-house machine by relying entirely on a computer network to provide services was met with fear and loathing in most places. But since then Harvard and the University of Illinois' Center for Advanced Computation have shown that this approach is not only workable, but more economical. (It is difficult to buy or rent one machine that will satisfy the needs of all users. Among the machines in a moderate-size, heterogeneous network, it is fairly simple to find one that does satisfy any particular need.) For this situation a system is required that can handle a large number of terminals and provide the necessary network software to allow those terminals to be used over the network. Quite often additional devices are attached, such as line printers, special graphics devices, etc.

The primary advantages of front-ends or network access systems based on mini-computers are that they are inexpensive (quite often less than \$50,000), more flexible than larger machines (mini-computers are generally easier to interface to and write systems code for), and more reliable than large machines. Many mini-computers have less than one failure in 10,000 hours - or about 14 months - of continuous operation. The only real difference between

the front-end and the network access computer is that one of the "devices" connected to the mini-computer is the host. (The existence of such a "device" may, of course, significantly impact the other functions the front-end may perform for a given cost.)

Major Issues and Research

There are essentially two classes of problems that must be addressed in the design of front-end systems. First, there are the general problems of planning an operating system architecture that can serve some of the functions of a message switching system and some of the functions of a time-sharing system, and can also provide efficient access to remote resources. Second, there are the problems associated with the front-end interface. We will discuss the front-end problems first, and then the access system architectures that are being experimented with.

There are two basic approaches to front-ending: the special purpose approach and the general purpose approach. With the special purpose method, both the host and the front-end are intimately familiar with the other's inner workings. This usually means that the entire front-end system is not suitable as a front-end for a different machine (and often limits the networks to which it may be attached). In the general purpose method, the only part of the front-end system specific to the machine being front-ended is that part responsible for manipulating the hardware interface and interpreting the protocol between the two machines. Since the front-end has few constraints that it imposes (normally only some sort of flow control and a limitation on complexity), the major constraints are imposed by the host. These constraints, of course, are mediated by the variety of network services that are to be made available and the amount of freedom one has to modify the host's software.

Since the class of machines that are usually hosts are often very difficult to interface to, close attention must be given to the actual hardware access to the host. There are essentially three ways the front-end can appear to the host. It can look like (1) a terminal, (2) a disk or tape drive, or (3) another computer. In the case where the front-end appears to be a terminal or at best a multi-drop line, the front-end provides a relatively low speed interface into the host's terminal access system. (The terminal approach is quite often used where the network is to appear transparent to users of the host.) For some limited applications the terminal approach can be accomplished without the necessity of writing host software. If the front-end appears as a tape drive or disk to the host, this provides a much higher bandwidth into the host. The tape drive approach will require some software (but possibly not operating system modification) in the host. If the front-end appears as another machine, then quite high bandwidth may be achieved. This cooperating-machine approach will require, in most cases, modification of the operating system, as well as user-level utility packages. Although this approach allows the greatest bandwidth and the most flexible interface with the front-end, it is quite often impractical because of the cost of the software and the difficulties of attaching the two machines together.

Let us look at the range of modifications to the host software and its affects on the kinds of services that may be provided. If no software may be placed in the host to aid the front-end, then the services that may be provided are severely limited. Essentially the class of services that can be allowed in this case are those that may be provided in such a way that

the network can be made transparent and considered an inexpensive phone line. For example, in the ARPA environment this means that neither the File Transfer Protocol nor the Network RJE protocol could be implemented on the front-end. These protocols require that the front-end implementations be able to manipulate the host's file and remote job systems. The front-end in this case is limited to the host's interactive user command language as its only tool to manipulate the host system's resources. These commands and their responses are usually oriented too much towards human use to be suitable for machine generation and interpretation.

To have the best of all worlds one would like to interface the front-end via a high speed interface and have local software implement the high-level network protocols, such as remote job entry or file transfer. However, the degree to which this approach may be taken is often limited by non-technical constraints, such as lack of manpower, expertise, or the legalities of maintenance contracts. Most operating systems do not allow user-level programs to provide their own device drivers. These functions must be done deep in the bowels of the operating system. In many instances this is impractical and undesirable. This limitation usually means that the front-end must be made to look like a device a user program can access.

Once these problems have been addressed it is necessary to consider how the front-end and the host are going to discuss the network. There are two basic approaches: (1) The front-end handles only the communication-specific aspects (the low-level protocols), and the host is provided a simple set of primitives to establish associations and send data. In this case the host implements the higher level protocols. (2) The front-end knows about all protocols, implements them, and provides a small set of primitives to the host to perform the necessary functions (cf. [Padlipsky, 1974]). It is not clear which approach is the most efficient or flexible. The second does have the advantage that any other devices attached to the front-end may make use of the front-end protocol implementations to gain access to the net. For example, a printer attached to the front-end could have listings of files transferred to it, by using the network's file transfer mechanism and without impacting the host. Since minicomputer operating systems are less complex and therefore generally less expensive to modify, this approach also may make implementations of new or revised protocols more economical.

The most controversial issue associated with network-access computers is how much they should do. Consider the extremes: One extreme view is that the computer should do as little as possible. The access system should provide basic network support and support a moderate class of devices such as terminals, printers, etc. Persons holding this viewpoint might draw the line at local file systems or rotating storage. At the other extreme, one might argue that local facilities should include the ability to run programs on a limited basis. For example, a problem at the level of balancing a check book should be done locally; an inventory, however, should probably be handled elsewhere. As usual there have as yet been no real cost studies done so it is unknown exactly what the trade-offs are. Advantages of providing more facilities are that the user may have access to files in the event of a host or network failure, have closer control on archiving, and may write local utilities to manipulate the network. In addition, the hardware used for such systems is usually under-utilized, and could support additional functions with little increase

in cost. The degree to which the second opinion is subscribed to over the first depends greatly on the kind of access the user community requires and how much it can afford.

The present tendency has been toward providing more local facilities. Once this route has been chosen, the next major problem to be addressed is how to organize an operating system that is half message-switch and half timesharing system. This is a very difficult problem and as yet there is not enough experience with the requirements of the users of such systems and the constraints of various designs to draw many conclusions.

Several groups have been experimenting with possible network access system architectures. One group associated with the Speech Communication Research Lab (SCRL) in Santa Barbara has been developing a system called ELF that runs on the PDP-11 [Retz, 1973]. This system breaks little or no new ground in the world of operating system design and is structured much the same as a conventional time-sharing system.

Two other groups (at SUNY Stony Brook and U of I CAC) have taken a more radical approach. Their systems are based on what has been referred to as a thin-line approach. Traditionally, an operating system distinguishes between accessing local and remote resources. More efficient but machine-dependent "thick-line" (i.e. small time delay) techniques are used to access local resources, while less efficient machine-independent "thin line" (i.e. relatively long time delay) procedures are used to access remote resources. The premise is that a "thin-line" approach to operating system design will be more modular and easier to modify, and allow more efficient remote access with slightly less efficient local access than a system with two separate techniques. Since users of a network-access system will be accessing remote resources more often than local ones, this approach should be advantageous. The SUNY system [Akkoyunlu et al., 1972] is based on the ports concept [Balzer, 1971] and a message-switch-oriented interprocess communication system [Walden, 1972]. As yet no information is available as to what the operating characteristics of this system are.

The U of I effort, ANTS, is based on a stylized finite-state machine form of a process and a line-switch protocol [Bouknight, 1973]. This system proved to be somewhat unwieldy primarily because of the approach taken to memory management and the high overhead incurred by the internal protocols for the movement of data. Unfortunately little or no experimentation was possible to determine the extent to which these problems could be solved. It may be interesting to note that the ANTS Network Control Program was the smallest in the ARPA network, the next largest being two and a half times greater [Postel, 1974].

Summary and Assessment

The present knowledge about front-ends and network-access machines is slight. At present there is little experience with both networks and access systems since only a few attempts have been made to build such systems. For those that are being built, it is too early to have any data to evaluate them. The basic problems are that it is unclear what a network access system should be capable of providing for its users, and that a better understanding and evaluation of various high-level network protocols is needed to determine

the necessary network facilities that should be provided (This kind of protocol work often sheds light on general concepts for organizing systems.) As more work in these areas and data from the performance evaluation of the existing systems becomes available it will be possible to propose and develop better network access systems.

References

- Akkoyunlu, E.R., Bernstein, A. and Schantz, R.
1972 "An Operation System for a Network Environment", Technical Report No. 5, Dept. Computer Science, SUNY Stonybrook, N.Y.
- Balzer, R.
1971 "Ports - A Method for Dynamic Interprogram Communication and Job Control", AFIPS Conference Proc. 40, pp. 485-489.
- Bouknight, W.J., Grossman, G. and Grothe, D.
1973 "The ARPA Network Terminal System: A New Approach to Network Access", Data Networks: Analysis and Design, 3rd Data Communications Symp., IEEE, pp. 73-79.
- Padlipsky, M.A.
1974 "A Proposed Protocol for Connecting Host Computers to ARPA-Like Networks via Front-End Processors", RFC #672.
- Postel, J.
1974 "Survey of Network Control Programs in the ARPA Computer Network", MITRE Tech. Rpt. MTR-6722, MITRE Corp., McLean, Va.
- Retz, D.
1973 "Elf Documentation", Internal memo, Speech Communication Research Lab, Santa Barbara, Ca.
- Walden, D.
1972 "A System for Interprocess Communication in a Resource Sharing Computer Network", CACM 15, pp. 221-230.

Introduction

A great deal has been written regarding the legal and social implications of protection and privacy in computer systems. In keeping with the bulk of this literature we will define privacy to be a personal issue; protection to be the technological means of insuring privacy; and security to be the collection of policies which govern the application of a protection mechanism. In this discussion, we will focus entirely on protection mechanisms and their associated security policies.

Security is a pervasive consideration in the development of a resource sharing system. If any single component of the system can be compromised, then the system as a whole can be penetrated. An operating system may be secured and provide adequate protection mechanisms to implement a data management system on top of it. If, however, the data management system incorrectly implements security policies, then it can be penetrated regardless of the correct functioning of the operating system. The overall security of a system requires the secure functioning of the support personnel, the hardware, the system software, and the application software. This is why security is frequently called a weak-link phenomena. The whole system is only as secure as its least secured component. Figure 5, from Ware [1967], shows some of the system-wide security threats faced by a typical time-shared computer system. In a network, the number of potential weak links increases dramatically.

In the discussion that follows we will first consider the user authentication problem. The cornerstone of a good security system is in its verification that the user at a terminal is who he says he is. The classical approach to the authentication problem has been the typed alphanumeric password. There are serious deficiencies with this scheme. There are serious deficiencies with this scheme. Research into more sophisticated password schemes, including physiological measures of the user, hold some promise to remedy current authentication deficiencies.

Second is a discussion of data security and access control mechanisms. Providing data security alone does not insure system security. However, data security does close the largest and least expensive to penetrate security holes of most systems.

Third is a discussion of the far more subtle, and possibly insoluble, problem of confinement. The Trojan horse problem, the census problem, and the information leakage problem are each a related manifestation of the difficulty of confinement.

The fourth section discusses certification. The efforts of penetration teams and the nature of correctness proofs and security kernels are described.

The last section is a summary and assessment of the state-of-the-art in computer security.

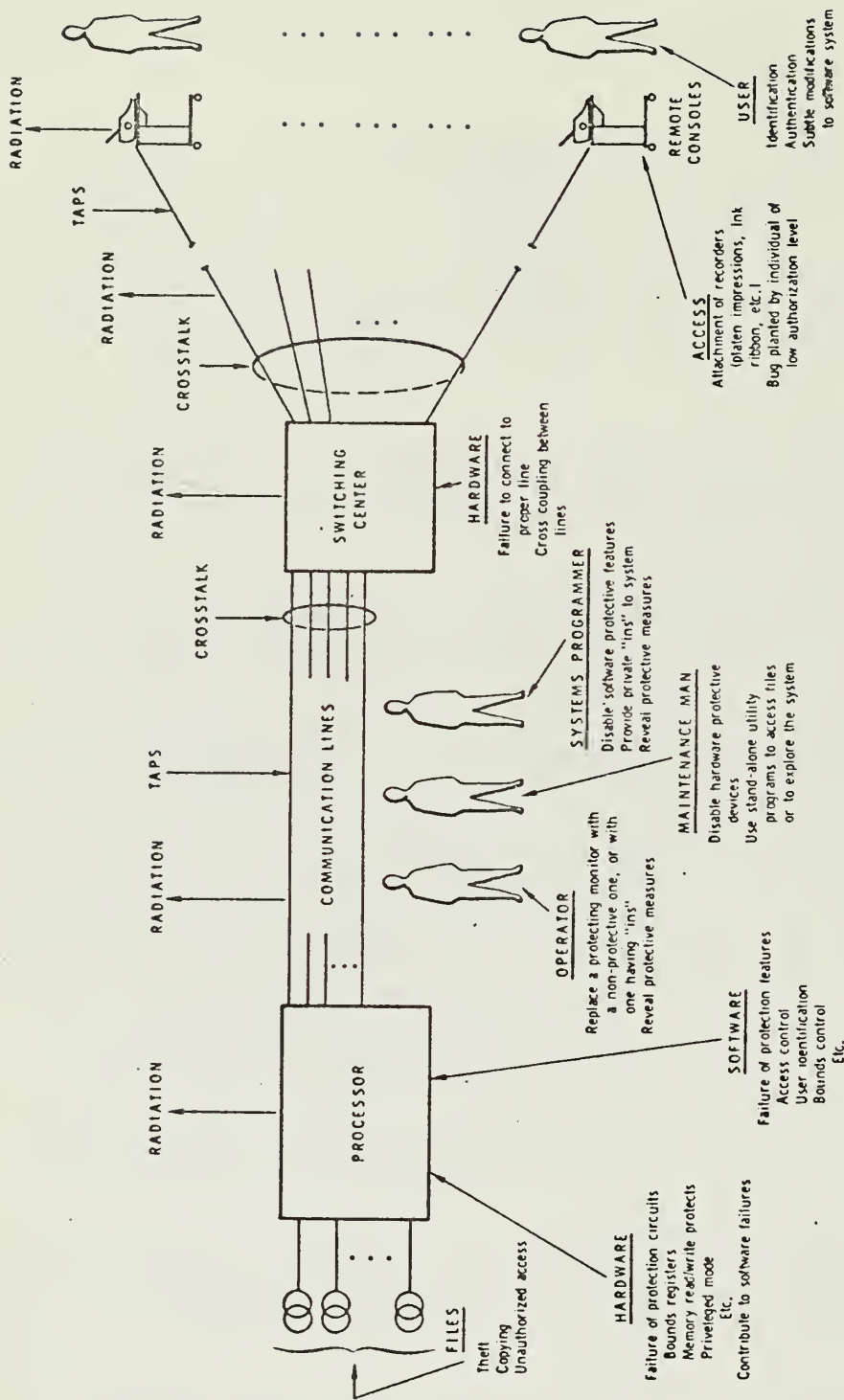


Figure 5 - Typical configuration of resource-sharing computer system. [Ware, 1967]

User Authentication

The authentication problem is the problem of verifying the identity of an individual user. In this context, this is the problem of generating and identifying a password which is unique to one individual. A password may be generated by a variety of means such as the typing of a unique alphanumeric codeword, the scanning of a thumbprint, the measurement of brain pulses, a sequence of finger motions on a touch sensitive display or a combination of these and other techniques.

Password Generation. In conventional systems, alphanumeric passwords are typed by the user to authenticate himself. The number of possible passwords must be sufficiently large to thwart disclosure attempts based on trial and error. Passwords must be periodically altered. The longer they remain static, the greater the possibility of compromise.

The generation of the password can be controlled by the system administrator or by the user. If the system administrator distributes a password, then both he and the user know it and it is no longer unique to the user. If the user generates the password, experience has shown that he frequently chooses a very short password or one easy to remember (e.g., his wife's name). In either case, the size of the password space is so small that trial and error or intelligent guesses are serious threats to compromise. On one system (Multics), the user can execute a command to produce a list of eight-character, pronounceable "random" passwords. This helps the user to choose a password which does not have a guessable relation to him and is long enough to thwart trial and error penetration.

A one-time password was described by Hoffman [1968] and also by Peters [1967] and implemented on ADEPT50 [Weissman, 1969]. The user is issued or generates a list of passwords (64 in ADEPT-50). Each password is valid once. Once it has been used, it is discarded. The user starts with the first password and proceeds down his list, crossing off each password as it is used. If the password is compromised after use, it is of no value since it is no longer usable. The technique has obvious problems of memorizing a long list and of compromise of the written list before password use. Compromise can at least be detected - the next password will have been used and will not be valid. The problem of memory can be assisted by a challenge-response one-time list. The user could supply a list of one-time passwords and prompts to solicit those passwords. (E.g., the challenge "dog" might solicit the password "cat", or "run", or "Fido", or "Rin-Tin-Tin", etc.) The password space is drastically reduced by this technique but the penetrator has only one guess. The user who generates the one-time list must be educated as to the need to choose obscure relations.

Password Input and Checking. There are several problems encountered at the time a user types his password. With the exception of one-time passwords, communication line security is required between the user's terminal and his host computer to prevent password theft

(obviously, a non-secure line can present many additional problems even for the one-time password). Some means must be provided to hide the password when it is typed. This is normally done by overprinting and thus blackening an area before typing the password or by not printing the password as it is typed (noprint on some half-duplex terminals and turn off character echo at full-duplex terminals). There are many difficulties with even this simple operation which further indicates the subtleties of the problem. For example, if a password is overprinted after it is typed, rather than before, then the over-the-shoulder observer can compromise it before it is blacked out. On most CRT terminals, the black out doesn't work because characters cannot be overprinted. If passwords are checked a character at a time and immediate notification given on error (as some early TENEX systems did) then the penetrator can trial and error test each successive character to compromise the password. If there is no delay in or limit to retries, then a computer could attempt a trial and error penetration at high speed. If user names must be accepted before a password can be entered, then information can be released about who the registered users of the system are.

One interesting technique to help check passwords on poorly secured machines is to store a transformation of the password in the password tables instead of the original password. Instead of checking the password directly, it is first transformed and the transformed value is checked with the tables. There are transforms that are not feasible to reverse. (E.g., see Purdy [1974] and Evans, et al. [1974].) Even if the penetrator knows the transformed password stored in the tables and the transforming function, he cannot work backwards to discover the original password which he must input to authenticate himself.

Password Give Away. If the security mechanisms in a system are clumsy, then the users of those systems will find it easier to give their passwords to their colleagues than to overcome the obstacle of a security system with poor human engineering. Virtually all current systems suffer from this problem. The solution is two-fold: better human interfaces and passwords that cannot be "loaned" are required.

Continuous Authentication. Once a user logs in and initially authenticates himself, checks must be made to verify that he hasn't left his terminal and been replaced by another person. One thought is to physically connect the user to the terminal and to send a signal if the connection is broken. The user must then re-authenticate himself when he reconnects.

Physiological Passwords. The need for the unloanable password and for continuous authentication has led to proposals to include physiological identifiers in authentication. These include finger prints, lip prints, voice prints, and a host of polygraph-like measures. One of the more interesting authentication proposals is to use event related brain pulses (ERP) for both initial and continuous authentication, possibly augmented by conventional alphanumeric codewords [Donchin, 1975]. The concept has the favorable attributes of a challenge-response one-time password (without the user memorizing lists), unloanable password, and continuous authentication. The concept is in the research stage to determine its feasibility.

Data Security

The concept of data security is simple: It is the assurance that no person (or program) can access any data except those items to which he has explicitly been granted access. This section will describe some of the access control mechanisms used by current operating systems to provide data security. Before proceeding, some terminology, which will be used throughout the remainder of this section, must be defined. An object is any identifiable construct which may be protected. Some examples of objects are files, segments, and main memory. A subject is an active agent performing in behalf of a user. Subjects manipulate objects in ways specified by the protection mechanism of the system, and may themselves be objects. An example of a subject is an executing process.

Capabilities. A great deal of the current research into access control mechanisms centers around the notion of capabilities, introduced by Dennis and Van Horn [1966]. In their original model, Dennis and Van Horn used capabilities as a means of determining what accesses were to be permitted to a given subject. The subjects in their model are a group of cooperating processes, called a "computation"; each computation is executing in a "sphere of protection" specified by the list of capabilities associated with the computation. Each of the capabilities associated with a computation is used simultaneously to locate an object and to specify what access the computation is to have with regard to that object.

The following is a convenient way of viewing a capability-based protection model. Imagine an access matrix, A , whose rows are identified by subjects, and whose columns are identified by objects. The entry $A[S,O]$ contains a complete description of the ways subject S may access object O . That is, it contains S 's capabilities with respect to O . To determine if S has some capability, c , with respect to O , it is only necessary to inspect $A[S,O]$ for the presence of the capability c . This view of capabilities is due to Graham and Denning [1972], and is used by them to suggest some necessary capabilities and a set of primitives to manipulate both the individual capabilities and the access matrix itself.

Domains. In their original model, Dennis and Van Horn postulated a sphere of protection that defined the capabilities of a group of cooperating processes. A simplification of this concept by Lampson [1969] has extended the scope of the capability models. Lampson establishes a new type of object, which he calls a domain, used solely to group capabilities. Each process is executing in some domain at every point in time and can exercise only the capabilities which belong to that domain. When control passes from one domain to another, the capabilities of the process will change to become those of the new domain. Switching domains is clearly a sensitive action, and is permitted only under well-defined circumstances.

If Lampson's domain approach were to be implemented by means of an access matrix as described above, the matrix would expand to three dimensions. Thus, to locate the capabilities of subject S with respect to object O while S is executing in domain D_i , examine the access matrix at $A[S, D_i, O]$.

Implementations. There are three straightforward ways to implement a protection mechanism based on capabilities: an access matrix, capability lists (C-lists), and access control lists (ACL's). The most obvious implementation is the access matrix scheme described above. This method has one major drawback: a matrix of all the objects and all the subjects in a modern computer system would be enormous. Such a matrix would also be sparse, wasteful of storage space and costly to search, since most subjects would have a system-defined default capability with regard to most objects. The following paragraphs describe some alternative methods of implementing the access matrix.

Capability Lists. C-lists and ACLs both involve the partitioning of the access matrix. If the matrix is partitioned by rows (C-lists), then we can associate each row directly with the corresponding subject. This has the advantage of co-locating all the capabilities of a subject directly with that subject. The biggest drawback to this scheme is that it is very difficult to locate all the capabilities that refer to a given object. This can be a serious deficiency if it is necessary to locate everyone who has access to a given file, as during a security audit. This approach has been called the capability-list method, and is described in Lampson [1971].

Access Control Lists. The ACL method for storing the protection matrix is the opposite of the C-list approach. Instead of associating the protection information with the subject, it is kept with each object. This corresponds to storing the relevant column of the access matrix with each object. If an entry for subject S is not found in this access control list, then S has the (presumably null) default capability with respect to the object associated with the ACL. This scheme makes it very easy to determine all subjects which have access to a given object, but not to find all the objects which a subject has access to. Access control lists are described in Lampson [1971]. For a discussion of one implementation, see M.I.T. [1973].

Password per Object. An alternative to access control methods based on Graham and Denning's access matrix is to associate a password with each object. In order to gain access to the object it is necessary to supply the correct password. The drawbacks to this system are obvious. Each person that needs access to a given object (e.g., a file) will have to supply the password for that object and for every intermediate object needed to locate it (e.g., directories). The number of passwords needed to perform even relatively simple actions can be very large. In fact, the inconvenience suffered for the sake of protection may become greater than the benefits; a plausible user response to such a situation is simply to refuse to bother with the plethora of passwords. In this fashion, a workable protection facility may go substantially unused due to the large burden that it imposes on its human users. It is worth noting that at least one major contemporary operating system (GCOS) uses precisely this scheme.

Rings. By the late 1950's a number of computer systems were using the techniques of multiprogramming to support several simultaneous users. One feature of such systems was a supervisor program that provided services such as I/O to user programs. In general, the supervisor had permission to execute certain instructions denied to the user programs; the hardware of the system enforced this by disabling these instructions when the system was in "user mode". The concept of domains introduced by Lampson (see above) is a generalization of this technique. Lampson defines a domain simply as a context in which a process finds itself [Lampson, 1969]; in the multiprogramming systems two such contexts were provided by the hardware.

A further generalization of this mechanism has been used in Multics [Schroeder and Saltzer, 1972]. Here, a series of hardware-enforced domains of successively greater privilege, called rings, is proposed. The only way to enter a ring of greater privilege than the current ring is via procedure entry points that have been designated as gates between rings. Any attempt to cross ring boundaries without using a gate will be rejected by the hardware.

Confinement/Leakage

The problem of insuring that an arbitrary process or program will not transmit information to an outside agent (i.e., another program or process) unless authorized to do so is called the confinement problem. At its most basic level, the problem is one of guaranteeing that there is no communication channel between two processes. We shall examine several methods of passing information via covert communications channels that circumvent access control mechanisms, and the difficulty of blocking those channels.

Information Leakage. Some examples of possible channels for information leakage may help to increase reader sensitivity to the confinement problem. Some potential channels are: 1) A program with memory (such as PL/1 static or Algol own variables) which can collect data until called to transfer it; 2) a utility which writes illicit information into a file in some known directory; 3) the system inter-process communication facility; and 4) creating contention for some known resource (e.g. tape buffers). The delay introduced in this fashion can be detected by other processes and used to slowly leak information.

It is apparent that these methods all make use of legitimate facilities of the operating system to pass information. The problem is how to block the illicit transmittal of information without forcing users to stop using all the functions of the operating system which could potentially be used as information channels. For a more complete treatment of the previous examples, along with many others, see Lampson [1973] and Popek and Kline [1974].

Trojan Horse. The classical method of establishing surreptitious information channels is the "Trojan Horse". In this technique, an ostensibly trustworthy programmer supplies a routine which, in addition to performing its stated function, also creates a data channel it can use to transmit information back to the programmer. The actual method

used is not important; what is important is that once the Trojan Horse routine has been invoked, the invoking agent is completely at its mercy.

An interesting complication to this problem is the case where neither the owner of the called function nor the owner of the calling routine trust each other. Consider the case of a business that has a proprietary software package, and a rival business that wishes to analyze proprietary data using the software package. The first firm wants to be sure that a user of their service cannot copy it, while the user of the package needs to know that it will not steal a copy of the data it is operating upon. Each of these mutually suspicious systems needs to be protected from the other; the capability and domain based protection systems described above cannot completely guarantee this. Schroeder [1972] proposes a mechanism to permit the cooperation of mutually suspicious systems.

The Census Problem. Another problem that falls under the heading of information leakage is the so-called census problem. This is the problem of inferring information from data, and is a cause for concern even given a well functioning data security mechanism. If a data management system allows the user to formulate queries, it is possible for a knowledgeable user (perhaps with some data from outside the system) to extract more information from the response to a query than the system designer intended.

For example, consider a data management system that operates on census data. To protect citizens no queries may be made regarding individual information, and no "sensitive" information (e.g., salary, standard of living) will be reported except as average values. If someone wanted to discover his neighbor's salary directly the system would not allow it. However, if he could formulate a query regarding average salaries with constraints so rigid that only he and his neighbor fit the qualification, he can infer the neighbors salary since he knows his own. Thus, without violating the security considerations of the system, he will have violated his neighbor's privacy. It is not clear that inferred data leakage can be prevented in the general case.

Encryption Techniques

The difficulty of guaranteeing the security of a data base - i.e. guaranteeing that no unauthorized person can see a file that he shouldn't - leads one to consider encoding the information to make it unreadable by outsiders. While cryptology is an ancient subject, it is only recently that serious, systematic study of its application to computer data privacy has been undertaken. Turn [1973] has prepared a good, comprehensive review of work up to 1973. We have relied heavily on that review in the preparation of this brief summary.

Basically, there are two main types of encryption techniques (or privacy transformations, as they are often called). These types are irreversible and reversible. Irreversible transforms are those from which the original data can not be recovered. Such transforms include the addition of random numbers to numerical data and averaging or otherwise aggregating the data. The idea is that if the data are to be used only for statistical purposes, transforms which preserve statistical properties

can be used to maintain privacy of the individual records. Irreversible transforms are frequently applied to research data in the social sciences, for example. To improve understanding of the technique, Reed [1973] has used information theory to analyze record distortion (formally defined) caused by various of these privacy transformations.

Reversible transformations include codes in the usual sense. Data compression schemes (cf. Compression section of this report) come under this heading. The term coding is usually applied to transformation of entire words (or other relatively large items) into other words e.g., into another language. Encoding or decoding usually requires the use of a dictionary; the problem then is to make the dictionary secure. Substitution usually refers to a letter-by-letter transformation. The familiar cryptogram is an example of this. Transposition transformations permute the ordering of letters or numbers and may be carried out by simple systematic techniques. Finally, one can use a combination of methods to complicate the task of breaking the code.

The choice of an encryption technique depends on several factors. First of all there is obviously the effectiveness of the encoding. Effectiveness from the information theoretic point of view has been studied by a number of researchers, from Shannon [1949] to Turn [1973]. Another approach to studying effectiveness has been to try and determine how much computational effort is expected to be necessary to break a given code. For an example of this work see [Tuckerman, 1970]. A second important consideration in choosing a privacy transformation is cost. This is usually not a serious problem, since it takes only a very simple piece of hardware or software to carry out any of the commonly used encryption schemes.

When data is to be communicated via media which are not secure, such as telephone lines, some form of encryption is normally used to increase the cost to violate the security of the communication. A very straightforward approach is taken in conventional "scramblers" [Davies and Barber, 1973]. Data to be sent through a communication line is transformed, usually on a bit-by-bit basis, by exclusive-or'ing each data bit with some invertible logical function of the previous n bits. The value of the parameter n directly affects the frequency at which the resulting series of bits repeats (given constant data), and thus determines the cost to break the code. The unscrambler at the receiving end simply uses the inverse of the encoding function to translate the data back into its original form. This simple scheme would thwart only casual penetrators.

Baran [1964] uses the notion of keys to produce encryptions which are dependent upon both the data being sent and on a key presumably known only to the sender and receiver. The key may, in fact, be a number generated by a pseudo-random number generator synchronized at the two ends and started from the same initial key. This solution depends on the physical security of the encryption/decryption equipment and on the secrecy of the keys themselves. In a computer network, where the data may pass between many switching nodes in getting to its final destination (c.f., Communications and Networks), the data would be

available in clear (unencrypted) form at each node. Since the violation of only one node would endanger all communications passing through the node, Baran also suggests end-to-end encryption. Using this scheme, the actual data can be sent through the network encrypted, with only the ultimate destination available in clear form at the switching nodes. By making the end-to-end encryption scheme depend upon all previous data, the fact that networks route different parts of the same logical stream of data to their destination via different paths can be used to advantage. If a security violator who has somehow obtained the initial keys and the correct algorithms misses one piece of the stream of data, he must re-synchronize his decryption devices to the rest of the stream. This can be made extremely difficult to do.

Certification

For an operating system to be certified secure, someone with appropriate authority must be willing to sign a document saying it is secure. Such decisions are not made lightly. At the present state of secure system technology, no one has been willing to sign for a multi user system.

The Cost to Break the Security of a System. Any system can be broken, given sufficient capital or appropriate influence. Even if the system is technically secure, other channels exist. Most system programmers would at least consider making a mistake - a very carefully planned mistake - for a million dollars and a promise not to harm his family. Even if such channels failed, the system breakers could find still more devious routes - such as kidnapping the Secretary of State and demanding the secured data they want as ransom, or stealing a pound of plutonium and threatening to poison New York City's water supply. The examples are obviously outrageous fantasies. They are intended only to affirm the point that the only real measure of security is the cost to break it. if the cost to penetrate is greater than the expected return, an agent will not find the penetration profitable.

In the cost-to-break range that most operating systems are found, less esoteric methods are necessary. To guess a password, for example, a small computer costing \$20,000 simulating a user terminal and a few weeks of work would suffice on some systems. More typically, a few hundred dollars worth of computer time or a few hours reading of reference manuals might be all that was required. The important point is that any system has a weakest link, and given sufficient capital, the system can be broken. The job of the secure system designer is to make the weakest link in his unprofitable to exploit (Table 1 [Petersen and Turn, 1967]).

Secure System Technology. A classic method used to discover security flaws in supposedly secure systems is to use teams of penetration experts. These experts try to penetrate the system by any methods possible. Such teams have invariably been successful [Popek, 1974], [Popek and Kline, 1974]. The methods used are primarily heuristic, and the use of penetration teams, while successful from the standpoint of

TABLE 1
[Peterson and Turn, 1967]

SUMMARY OF COUNTERMEASURES TO THREAT TO INFORMATION PRIVACY

Threat \ Countermeasure	Access Control (passwords, authentication, authorization)	Processing Restrictions (storage, protect, privileged operations)	Privacy Transformations	Threat Monitoring (audits, logs)	Integrity Management (hardware, software, personnel)
Accidental: User error	Good protection unless the error produces correct password	Reduce susceptibility	No protection if depend on password; otherwise, good protection	Identifies the "accident prone"; provides <u>post facto</u> knowledge of possible loss	Not applicable
System Error	Good protection, unless bypassed due to error	Reduce susceptibility	Good protection in case of communication system switching errors	May help in diagnosis or provide <u>post facto</u> knowledge	Minimizes possibilities for accidents
Deliberate, passive: Electromagnetic pick-up	No protection	No protection	Reduces susceptibility; work factor determines the amount of protection	No protection	Reduces susceptibility
Wiretapping	No protection	No protection	Reduces susceptibility; work factor determines the amount of protection	No protection	If applied to communi- cation circuits may reduce susceptibility
Waste Basket	Not applicable	Not applicable	Not applicable	Not applicable	Proper disposal procedures
Deliberate, active: "Browsing"	Good protection (may make mas- querading neces- sary)	Reduces ease to obtain desired information	Good protection	Identifies unsuc- cessful attempts; may provide <u>post facto</u> knowledge or operate real-time alarms	Aides other counter- measures
"Masquerading"	Must know au- thenticating passwords (work factor to obtain these)	Reduces ease to obtain desired information	No protection if depends on password; otherwise sufficient	Identifies unsuc- cessful attempts; may provide <u>post facto</u> knowledge or operate real- time alarms	Makes harder to ob- tain information for masquerading; since masquerading is de- ception, may inhibit browsers
"Between lines" entry	No protection unless used for every message	Limits the infiltrator to the same potential as the user whose line he shares	Good protection if privacy transformation changed in less time than required by work factor	<u>Post facto</u> analysis of activity may provide knowledge of possible loss	Communication net- work integrity helps
"Piggy-back" entry	No protection but reverse (processor-to- user) authenti- cation may help	Limits the infiltrator to the same potential as the user whose line he shares	Good protection if privacy transformation changes in less time than required by work factor	<u>Post facto</u> analysis of activity may provide knowledge of possible loss	Communication net- work integrity helps
Entry by system personnel	May have to masquerade	Reduces ease of ob- taining desired infor- mation	Work factor, unless depend on password and masquerading is successful	<u>Post facto</u> analysis of activity may provide knowledge of possible loss	Key to the entire privacy protection system
Entry via "trap doors"	No protection	Probably no protection	Work factor, unless access to keys obtained	Possible alarms, <u>post facto</u> analysis	Protection through initial verification and subsequent main- tenance of hardware and software integrity
Core dumping to get residual information	No protection	Erase private core areas at swapping time	No protection unless encoded processing feasible	Possible alarms, <u>post facto</u> analysis	Not applicable
Physical acquisition of removable files	Not applicable	Not applicable	Work factor, unless access to keys obtained	<u>Post facto</u> knowledge form audits of personnel movements	Physical preventa- tive measures and devices

finding flaws, can not prove a system secure. A penetration team can only point out failures of system security; they cannot assert its success. What is needed is a way to prove that a system is secure, under a suitable set of restrictions.

Provably Secure Systems. Some method of producing systems which can be rigorously proven to correctly perform their security function is needed. Several attempts to produce secure operating systems have been made (e.g., [Weissman, 1969], [Saltzer, 1974], [Hoffman, 1973]), but only recently have serious attempts been made to mathematically prove the correctness of an operating system. There are several reasons that this situation exists. The most important is the inherent difficulty in proving large programs and the size and complexity of existing operating systems. For example, the largest program proven to date is 2000 lines long [Popek and Kline, 1975]. Furthermore programs in excess of a few lines must be very carefully designed to be proven using current proof techniques; even small operating systems written without an eventual proof as a major design influence simply cannot be proven correct at present.

Security Kernels. To facilitate the proof of correctness of an operating system, current researchers are isolating the security functions of the system and placing them in a security kernel. A security kernel contains only the security-relevant portions of the system primitive operations. By isolating this code and building upon it, it is possible to prove the security code to be correct, independently of the rest of the operating system. Although it is possible to introduce flaws at higher levels, reasonable caution can prevent this. An important advantage gained by this approach is that the security of the system need not be rigorously reverified whenever a change is made to some non-protection aspect of it [Popek, 1975], [Popek and Kline, 1975], [Popek and Kline, 1974].

Popek [1975] discusses at great length the advantages and disadvantages of placing the security primitives at the very lowest level of the operating system. For example, he considers whether it would be more appropriate to place virtual memory management routines at a level lower than that of the security primitives. The arguments in favor of this scheme point out that a great deal of protection data is necessary in order to determine the validity of any given operation, and that if the virtual memory primitives are available to the security primitives then the security primitives have the advantage of not being forced to perform their own I/O. However, there are significant disadvantages to this scheme. First, the security kernel is now forced to depend upon the correct operation of a large body of code having nothing to do with security per se. The second major disadvantage is more subtle. If a security kernel is depending upon the virtual memory facility then it may be necessary to interrupt the security primitives at any point in order to perform paging. However, the proof of correctness of the security primitives is much simpler if it can be guaranteed that certain instruction sequences will not be interrupted. With the security primitives at the lowest level of the system, they can operate in a non-interruptable mode without detriment to the rest of the system.

Summary and Assessment

At this time there is a very large and active community engaged in research into various aspects of computer security. Of the major topics that this report has discussed, by far the greatest activity is in the field of data security. Although the basic paradigm used by the researchers in this area is nearly 10 years old a great deal of progress has been made. It is probable that some certifiable data secure systems will become operational in the near future.

The work being done on the confinement problem is not progressing as well. The confinement problem is considerably more difficult to solve than the data security question. Resolution of the problem cannot be expected soon; indeed, the problem may not be solvable. This is not as gloomy as it appears. The question of data leakage and confinement is not as great a concern as data security. To exploit the low-bandwidth leaks of data is very expensive, and unprofitable for most data.

Finally, the authentication issue. No current method of authenticating the identity of a remote user is reliable. The human interfaces to the authentication mechanism are clumsy; some seem to encourage violations rather than compliance. There is some hope that current research into physiologically-based continuous authentication schemes may prove fruitful on a medium-range time scale.

References

Baran, P.

- 1964 "On Distributed Communications: Security, Secrecy, and Tamper-Free Considerations", Rand Corporation Memorandum RM-3765-PR, The Rand Corporation, Santa Monica, Ca., Aug. 1964.

Davies, D.W., and Barber, D.L.

- 1973 Communication Networks for Computers, Wiley Series in Computing, John Wiley and Sons.

Dennis, J.B., and Van Horn, E.C.

- 1966 "Programming Semantics for Multiprogrammed Computations", CACM 9, pp. 143-155.

Donchin, E.

- 1975 Private Communication, Feb. 1975.

Evans, et al.

- 1974 "A User Authentication Scheme Not Requiring Secrecy in the Computer" CACM 17, pp. 435-442.

Fabry, R.S.

- 1973 "Dynamic Verification of Operating Systems Decisions", CACM 16, pp. 659-668.

- 1974 "Capability-Based Addressing", CACM 17, pp. 403-411.

Graham, G.S., and Denning, P.J.

- 1972 "Protection - Principles and Practice", AFIPS Conference Proceedings 40, pp. 417-429.

Hoffman, L.J.

- 1968 "Computers and Privacy: A Survey", Stanford Linear Accelerator Center, Stanford University, SLAC-PUB-479, August 1968. (indirect ref)

Hoffman, L.J.

- 1973 "IBM's Resource Security System (RSS)", in Security and Privacy in Computer Systems, L.J. Hoffman, (ed), Melville Publishing Co., Los Angeles, Ca.

Lampson, B.W.

- 1969 "Dynamic Protection Structures", AFIPS Conference Proceedings 35, pp. 27-38.

- 1971 "Protection", Proc. Fifth Princeton Symposium on Information Sciences and Systems, Princeton University, March 1971, pp. 437-443, reprinted in Operating Systems Review 8, January 1974, pp. 18-24.

- 1973 "A Note on the Confinement Problem", CACM 10, pp. 613-615.

Massachusetts Institute of Technology

1973 The Multics Programmer's Manual, Rev. 15, Part II, Sec. 3.4.

Morris, J.H.

1973 "Protection in Programming Language", CACM 16, pp. 15-21.

Peters, B.

1967 "Security Considerations in a Multi-programmed Computer System", AFIPS Conference Proceedings 30, pp. 283-286.

Petersen, H.E., and Turn, R.

1967 "System Implications of Information Privacy", AFIPS Conference Proceedings 30, pp. 291-300.

Popek, G.J.

1974 "Protection Structures", Computer, June 1974, pp. 22-31.

Popek, G.J., and Kline, C.S.

1974 "Verifiable Secure Operating System Software", AFIPS Conference Proceedings 43, pp. 145-151.

1975 "Verifiable Protection Systems", to be published in ACM/IEEE Software Reliability Conference Proceedings, April 1975.

Purdy, G.B.

1974 "A High Security Log-in Procedure", CACM 17, pp. 442-445.

Reed, I.S.

1973 "Information Theory and Privacy in Data Banks", AFIPS Conference Proceedings 42, pp. 581-587.

Saltzer, J.H.

1974 "Protection and the Control of Information Sharing in Multics", CACM 17, pp. 388-402.

Schroeder, M.D.

1974 Cooperation of Mutually Suspicious Subsystems in a Computer Utility, Massachusetts Institute of Technology, Project MAC Report MAC-TR-104, 1972.

Schroeder, M.D., and Saltzer, J.H.

1972 "A Hardware Architecture for Implementing Rings", CACM 15, pp. 157-170.

Shannon, C.

1949 "Communication Theory of Secrecy Systems", Bell System Tech. J. 28, pp. 654-715. (indirect reference)

Tuckerman, B.

1970 "A Study of the Vigenere - Vernam Single and Multiple Loop Enciphering Systems", IBM Research Report RC 2879, May 1970. (indirect reference)

Turn, R.

1973 "Privacy Transformations for Databank Systems", AFIPS Conference Proceedings 42, pp. 589-601.

Ware, W.H.

1967 "Security and Privacy in Computer Systems", AFIPS Conference Proceedings 30, pp. 279-282.

Weissman, C.

1969 "Security Controls in the ADEPT-50 Time-Sharing System", Proceedings of the FJCC, 1969, AFIPS, reprinted in Security and Privacy in Computer Systems, L.J. Hoffman, (ed), Melville Publishing Co., Los Angeles, Ca.

Network Application Support

Overview

This section surveys current research and discusses the major problems imposed on computer applications by geographically distributed networks. The discussion is organized in three parts:

1. User Support,
2. Installation and Project Support, and
3. Network Support.

The state-of-the-art is most advanced in the area of user support. As a result of the ARPA network experience, many of the problems faced by a user have been identified, and, in some cases, solutions have been proposed and attempted. The principal problems for the remote network user involve documentation and consultation: (what is available? where? how do I use it? what do I do now that "your" program blew up on me?); communication with other users: network mail, terminal-to-terminal on-line messages, teleconferencing among collaborating remote personnel, and network information centers or clearing houses for collaborating groups; and user management facilities: setting up conferences, filing of correspondence, automatic memo/remainder services, personal abbreviations, and accounting services.

The network implications of installation and project support are not as advanced. The situation is similar for network support. In both of these cases the problems are more a question of policy development than of technology development.

User Support

Introduction

With the advent of the geographically distributed resource sharing computer network, the computer user is beset by problems that, although they always existed, could be dealt with by ad hoc or informal methods in the traditional computing environment. The network user may well find himself using computers, collaborating with other users, or seeking aid from consultants, none of which are within easy travel or phone distance from the user (and in fact are separated by several thousand miles). The user must therefore have available on the network a variety of support facilities. These facilities fall into three major categories:

- 1) System documentation and consulting.
- 2) User-to-user communication to aid collaboration among network users.
- 3) Management facilities to aid the user in keeping track of his far-flung projects on the network.

Since the difficulties may not be obvious to the uninitiated, we will, in what follows, discuss the major problems in each of these three categories as well as the major work in each area.

Documentation and Consulting

As anyone who has ever used or attempted to use a computer knows, having relevant, well-written and up-to-date documentation can mean the difference between an upset stomach and an ulcer. In the network environment, the situation is even more acute. The user may be sitting at a terminal a thousand miles or more away from the computer. He may not have hard copy documentation available to him. And he definitely doesn't know what has been going on at the service center (historically) that explains why he is having trouble. Normally, the local user who is having trouble physically goes to the service center to see a consultant, or to get the relevant documentation, or both. This is clearly not feasible for the network user.

A solution to the documentation problem that has been tried at several ARPANET hosts is based on a three-fold approach. First, the user is supplied with general overall information on how the system works. This information is in the form of various vendor and service center manuals. Second, the service center provides an organized corpus of on-line documentation, usually referred to as a help file. Third, the service center provides an on-line consultant that the user may ask questions of directly. There are difficulties with each of these partial solutions. As always, there is the problem of producing well-written documentation; however, this problem is further compounded in a network environment because of the colloquial nature of the jargon associated with computers. There is also the cost-effectiveness problem of what documentation should be provided to the user to save him from repeated requests for more documentation. Finally, the network on-line consultant must be able to field questions from a much broader range of users

than he would in a local environment. Traditionally, the consultant must answer the questions of the applications user who is more concerned with getting his answers than with learning the cleanest, cleverest method of obtaining them. The network consultant is questioned by the entire gamut of users from the applications user to the systems expert.

There have been some attempts by local systems to provide on-line documentation systems of their services. A good example is the DELPHI system developed at Case-Western Reserve University for their PDP-10 [Calvin, 1974(a)]. This system provides a bulletin board of consultant and machine schedules, announcements of new or updated software, and similar relevant day-to-day information. There is also a gripe mechanism by which a user may file a complaint or suggestion about a particular subsystem and have the complaint automatically forwarded to the responsible programmer and the system coordinator. When a new program or subsystem is added to the system, certain information is required by DELPHI. This information includes the name of the program, the responsible programmer, documentation telling how to use it, etc. DELPHI also provides commands for browsing and interrogating the data base of documentation and help text. DELPHI contains most of the necessary components of a help system; however, an on-line help system is only valuable if pertinent and up-to-date information is placed in it. This can only be accomplished by good systems administration. Figure 6 shows a survey of six on-line help systems on the ARPANET and the facilities they provide.

Network-Oriented Documentation Support. When the user decides that his needs can best be served by a resource sharing network, he is still confronted with several difficulties. The first of these is finding what services exist on the network, which are best suited (and most economical) for the problem at hand, and how the services selected are used. Since it has only been in the last few years that such resource sharing networks have begun to attract the non-expert (i.e., the non-systems programmer) the problems of the uninitiated are somewhat unspecified and have not been explored in any depth. There have been two efforts in this area, one attacking the problem of finding resources and the other providing a network-wide help system, a help-oriented protocol if you will.

A first approach to the problem of resource location has been made at MITRE [Benoit and Graf-Webster, 1974]. This consisted of a small data management system called REX with a data base listing various network services, their locations, and other relevant information. This system could be used to find a particular service for a class of machines and then to determine other information about the service, such as cost, local sources of expertise, etc. The authors of this system believe that such a facility could considerably improve a user's ability to manipulate the network.

There has been some discussion [Jernigan et al., 1973] and some definite proposals ([Calvin, Day et al., 1973] and [Sternick and Iseli, 1974]) for a network-wide help facility. The primary rationale for this approach is not to centralize all help information, but to provide a common method by which help systems, and through them users, could access various services' help information on other hosts. The proposals made thus far address the major issues, such as providing authoring capabilities for a user, scenarios and tutorials on services, keyword searching, etc. However, more detailed work is necessary to actually specify the mechanisms and requirements of such a system.

Host System Attributes	a	b	c	d	e	f
1-Pre-login capability (experimental accounts)		X	X	X	X	X
2-Experimental use of resources	X		X	X	X	
3-Structured Index	X	X			X	X
4-User Feedback Mechanism	X	X	X	X		
5-Hierarchical Structure	X	X	X	X	X	X
6-Primarily a Site Help Aid	X	X			X	X
7-Provision for On-line Help	X				X	
8-System Librarian Interface	X					
9-Adequately Self Describing	X				X	X
10-Keyword Search	X		X		X	

Attributes of Select Server Host Help Facilities
(evaluations are subjective on basis of attempts)

- a. DELPHI @ Case-10
- b. HELP @ UCLA-NMC
- c. INFO @ SU-AI
- d. ZOG @ CMU-10A
- e. HELP @ UCSD
- f. QUERY @ NIC

Reproduced from [Calvin, Day et al. 1973]

Figure 6

User-User Facilities

The advent of the computer network has also meant that good, usable human-to-human communication through the network is imperative. In a network, users must be able to talk to consultants; programmers collaborating on projects must be able to exchange ideas and programs easily; project and system managers must be able to handle accounts and other administrative tasks; etc.

The traditional techniques for human-human dialogue in a time-sharing environment have been of two major types, real-time message exchange in a conversational mode and mailboxes. Most versions of the first technique are very similar. The user types a command like "TO <usercode>" followed by his message. The message is delivered immediately to the addressee if he is logged on. There is one version that is somewhat different and deserves mention - the TENEX LINK command. This command links the terminals of two users so that anything typed on either terminal appears on both (but any action invoked by what is typed happens in the typer's domain). This technique can allow one user to show another user how to perform an operation, or show him the contents of a file. A variant of the LINK command, called ADVISE, also allows the advising user's typing to cause action in the advisee's domain. This can allow an on-line consultant to help a user perform some task. The major drawbacks to this are that it ties up both terminals (you can't do something else while waiting for a reply), and if both type at the same time the characters become intermixed. The mailbox facility is just that; it allows one user to send another a letter which is appended to the addressee's mailbox file, which he may read at anytime.

Both of these techniques can be used quite nicely in a network environment. In addition, two other facilities can prove useful. These are telconferencing and the existence of a network information center. Below we will discuss the state-of-the-art and the primary difficulties encountered in providing mail, teleconferencing, and an information clearinghouse in a network.

Mail. In the area of user support one of the most successful developments of the ARPANET has been network mail. This facility, built as part of the file transfer protocol, allows a textual file to be appended to a user's mailbox at a host where the user has an account. The use of such a system quickly led to the desire for more comprehensive facilities. One problem was that one might know a person but not know his usercode or where his mailbox was. This suggested a data base of network users and their mailbox addresses, a sort of on-line network address book. This solution, although workable, has problems such as accessability. Furthermore, at times a user may move his mailbox, because of changing projects, jobs, etc., and this would require mail to be forwarded [Kudlick, 1973].

Users also felt that it was necessary to indicate the urgency of a piece of mail so that a user might be contacted immediately upon receipt if he were on-line. Other desirable features are some indication of the length of the letter, and the ability to send a citation in place of a fairly lengthy piece of text, like a paper, so that the user can access it at will and at his own expense. In some cases the sender might also wish to know when the receiver read the mail or to have the mail system ask the recipient for an RSVP (i.e., an immediate reply) and automatically return it to the sender.

One environment in which most of these facilities have been provided or their primary aspects are addressed is represented by the proposed ARPANET mail protocol [White, 1973]. However, in attempting to provide a mail service to a network community, several technical problems must be overcome. These problems revolve around two major areas, security and accounting.

In the ARPANET, computer accounts are arranged directly between the consumer and the producer. If a user does not use a particular machine, then he does not need to have and probably doesn't have an account there. There has been some discussion of a network-wide accounting system, but there are many political and technical problems that need to be considered. Therefore, in this political atmosphere, who pays for the computer costs incurred at the destination host when mail is delivered? Clearly, the receiver does not want to be charged for unsolicited mail. Also, in keeping with a time-honored (but faltering) American tradition, mail should be inexpensive. The solution arrived at in the ARPANET was that mail delivery costs would be small and should be absorbed in system overhead and so "free" to the user. Those interested in the particulars of the discussion should see [Padlipsky, 1973 (a), (b)], [Bressler, 1973] and [Pogran, 1973]. All indications to date seem to justify this solution.

One of the major points brought up in the course of arriving at the above solution was that "free" is not the same thing as not requiring a log-in, and that even for sending mail the security of the receiving host must be maintained. To accomplish this, hosts may require a published mail usercode, which then restricts that user to mail delivery alone [Pogran, 1973].

There is also a need to protect the recipient of mail from "prank" or "phantom" senders by verifying that a sender is indeed who he purports to be. Thomas [1974] has suggested a technique that can be used in a network where all hosts can control program access to associations between remote communicating programs. However, if one host can not exercise such control, as in the ARPANET, the method is easily compromised.

Similarly, there are problems in controlling access to mailbox files. To date there is no system on the ARPANET in which it is not a straightforward task to read another user's mailbox [Bhushan, 1973]. This problem is more correctly considered in the context of operating system security.

Teleconferencing. A surprisingly useful idea in human-human interaction through a computer has been teleconferencing. This technique has been used in several environments for collaboration between geographically dispersed people. Teleconferencing has been useful for meetings of administrators where a typescript of the proceedings is desired, for example in the ARPA community and the Office of Emergency Preparedness. It has been used for demonstrations or classes where the students and teacher were separated by several thousand miles. It has also been used by a geographically distributed group of programmers to collaborate on the design of a new system, as in the network-wide help system described above [Jernigan et al., 1973]. This same technique has been exploited by one research group to elicit and evaluate expert opinions with surprising success on problems

sufficiently complex that no one person could embody all of the needed knowledge and then to synthesize a solution from the opinions [Lipinski et al., 1972].

One would assume, though, that teleconferencing is much less effective with respect to both cost and information flow than a real, live conference. Turoff [1972] has shown that, surprisingly enough, this is not the case. Turoff shows that for typical speeds (typing .5 words/sec, speaking 2 w/s), teleconferencing is more effective for fourteen people or more. In terms of cost, the results are even more dramatic. If one assumes that the computer is costing \$7 per person per hour of conference and that the conferees are worth \$10/hour, the (if no travel to the meeting is required) teleconferencing costs less for nine or more persons; if travel is required, then only four are necessary to reach the crossover. These numbers are based on fairly unsophisticated teleconferencing systems that do little but control access to the floor and keep a type-script. Even greater benefits can be obtained by a better system such as that proposed in [Duffield et al., 1974] or implemented by Calvin [1974].

Advanced teleconferencing systems are intended to provide much more versatility to the conferees and the chairman. For example, they provide automated aids for convening a conference; they enforce various speaking protocols (e.g., Roberts Rules, or round robin) and they provide the ability to pass the chairmanship of the conference and regain it, to control admission of new members, and to establish subconferences. For the conferees, facilities are provided for voting and access to scratch pads or to the computer for on the spot analysis or modeling, or for retrieving information relevant to a discussion. Private communication with the chairman is also allowed [Duffield et al., 1974]. Several of these techniques are provided in a network oriented teleconferencing system developed at CASE by Jim Calvin [1974]. Also, the PLATO project at the University of Illinois has developed a local teleconferencer that can maintain several conversations in view at once. This capability is possible since a homogeneous terminal population can be assumed. At present, it appears that there are no major theoretical hurdles facing teleconferencing, but more experience is needed in using and implementing these facilities.

A Network Information Center. One facility established early in the ARPANET was the Network Information Center (NIC) at SRI's Augmentation Research Center. The purpose of this center was to act as a clearinghouse and coordinator for various collaborating groups. The NIC maintains a directory of network participants, provides the distribution of Requests for Comments (RFC's), and aids specialized groups in their collaboration by providing the means for correspondence within the various groups.

This facility is very useful and provides the necessary focal point for several network-wide services, network documentation and protocol development. It also serves as a depository for evaluation of network services. In the ARPANET, a NIC was considered very useful; however, with the influx of less sophisticated users to the net it is only lately that the NIC's use has become critical. There is much remaining to be learned about what kinds of services can be provided and how they should be provided to best mesh with the user.

User Management Facilities

The remote computer user has working patterns that are radically different from those of a traditional computer user. This difference is usually manifested in the kinds of utility programs employed and is particularly critical for the user who is managing a complex system of software. The network user needs to be able to file network correspondence, along with his own relevant notes. He needs the ability to request a conference and have it set up for him [Calvin, 1974 (b)]. A prime consideration is for the user to be able to get up-to-date accounting information, while his privacy is maintained.

At present, the only system to provide such facilities at all is Multics [Multics, 1973 (a)]. Multics provides a number of commands that allow the user to get up-to-date accounting and to control the rate of spending; to set up personal abbreviations for commands, pathnames, etc.; to encrypt files; and to prepare documents. There is even a memo command that will remind one of appointments, tasks, etc. At least two groups have discussed providing such facilities to a network community. Strangely enough, both are unaware of the existing facilities on Multics.

Summary and Assessment

Computer networks have only become available to what could be considered the "normal" user community in the last two to three years. Therefore there has been little time and less funding for researchers to investigate the special problems that networking creates for the normal user.

A few of the problem areas, primarily mail, direct user-to-user communication, and teleconferencing, are exceptions. Most of the major difficulties (with the notable exception of mail authentication) with mail systems seem to be solved, although some may be discovered when implementations of White's proposed mail protocol [White, 1973] are attempted. Teleconferencing systems seem to represent no difficulties with the possible exception of the human engineering of the user interface.

On the other hand, network help facilities, user management facilities, and the functions of network information centers are poorly understood. Many of the problems of providing network users with acceptable and usable documentation support do not have technical solutions and require sensitizing the local service center staff to the world of the very remote user. For example, the traditionally colloquial nature of vendor-generated jargon can cause an explanation intelligible to a local user to be totally incomprehensible to a remote user familiar with a different dialect. The fact that the same words may be used to connote similar objects with widely disparate constraints can cause endless confusion to the reader who may not be aware of the writer's assumptions. There are also problems of maintaining the timeliness of on-line documentation. Also, some attention needs to be given to the human interface of such help systems. More work is also needed on the development of help protocols to provide network-wide documentation facilities.

At present, the problems relative to user management facilities are insufficiently understood to know exactly what facilities should be made available. More experimentation with the Multics-like facilities is needed. This experimentation may then lead to the generalization of some of these facilities to the networking environment.

Installation and Project Support

The management of a network installation or of projects using several network hosts requires new management tools. The system or installation manager needs facilities to allow the addition of new users, and the generation of bills on-line so they can be net-mailed to users. There must be on-line methods for handling rebates, and facilities for handling interactions with other installation and project managers for purposes such as negotiating costs, answering complaints or policy questions, etc. The project manager should be able to keep track, on a daily basis, of the state of his accounts and rate of spending of both the project and its members. It should also be possible for the manager to get graphs of spending. Some facilities to aid in projecting future use would also be helpful.

At present the only system to provide this kind of support is Multics. Multics has provided classes of commands that are useful to the system administrator and the project manager. The system administrator can add or delete users, run monthly bills, check log information, etc. The project manager, on the other hand, can check the amount of resources used by project members, set the rate of spending by project members, set preemption limits, etc. [Multics, 1973 (b), (c)].

Summary and Assessment

To date there has been no attempt to investigate the difficulties of providing similar facilities for multi-machine-based projects in a network. The areas of accounting and billing are of prime interest, especially the problem of setting up network banks. The problems of network accounting are especially difficult since they are heavily dependent on the political and financial relations in which the individual network sites are involved. In addition to the technical problems of maintaining a distributed accounting data base and assuring its security, very little is presently known about the variety of billing structures used by computer installations and what constraints they may have on network accounting and banking systems.

Introduction

In a network environment there are several necessary functions required for the support of the network as a whole. These can be broken into two major classes: The maintenance of the communication subnet and the evaluation of the network as a whole. This support requires both technical and administrative solutions. We will, in what follows, dwell mainly on the technical problems.

Recent Work

In a large computer network, it is necessary to detect and correct failures in the communication facility as quickly as possible. This has been accomplished in the ARPANET by establishing a Network Control Center (NCC) [McKenzie et al., 1972]. This organization is charged with the responsibility of maintaining the network communications, the development and distribution of IMP software, and the collection of traffic and failure statistics. Some work is presently being done [Kimbleton, 1974] to extend the scope of network control or management centers. Kimbleton is considering the use of multiple control centers in a network. These centers collect data to predict network growth patterns, and evaluate network performance.

In addition to the evaluation of the communication subnet, it is also necessary to evaluate the performance of the various host level protocols and of host service. A good discussion of the major issues may be found in the proposal for an ARPANET Performance Measurement Lab [Padlipsky et al., 1974].

The primary functions of network protocol evaluation are to act as input to further development or redesign of protocols for cleaner or more efficient implementation and to verify that host implementations are indeed obeying protocol. To date there has been little or no concerted effort to evaluate protocols other than host-to-host.

Another aspect of network performance measurement is a sort of network consumerism. It is desirable to verify that host services work as advertised by the documentation and to measure the cost of the service for typical use. Studies along these lines could provide an important input into the kind of resource location system discussed above.

Summary and Assessment

Most of the problems of network control centers are fairly well understood or are being actively investigated by BBN or ISI. However, the difficulties of protocol and service evaluation are much less well understood and are of both a technical and a political nature. There are several questions that need to be investigated, such as: What parameters of protocols should be measured? Are there valid cross-machine measures or testing procedures? What constitutes certification of a protocol implementation or host service? Obtaining answers to these questions will require considerable effort.

References

- Benoit, J. and Graf-Webster, E.
1974 "REX - A Resource Location and Acquisition Service for the ARPA Computer Net", MITRE Tech. Rpt. 387, MITRE Corp., McLean, Va.
- Bhushan, A.K.
1973 "FTP and Network Mail System", RFC #475.
- Bressler, R.
1973 "Free File Transfer", RFC #487.
- Calvin, J.
1974 (a) Delphi System Documentation; (privately communicated).

(b) "The Design and Implementation of an Interactive Teleconferencing Environment", Undergraduate Thesis, Case Western Reserve University.
- Calvin, J., Day, J., Greer, C., Hill, A., Iseli, J., Jernigan, M., Rogers, J. and Rosenfeld A.
1973 "A Stimulant for the Development and Implementation of a Help Information Protocol", Private Communication.
- Calvin, J., Iseli, J. and Jernigan, M.
1973 "Distributed Office Management System", Personal Comm., Sept. 1973.
- Duffield, H.C., Forman, E.H. and Iseli, J.
1974 "Capability Description of a Digital Teleconferencer", MITRE Working Paper 10813, MITRE Corp., McLean, Va.
- Jernigan, M., Iseli, J., Calvin, J., Hill, A., Greer, C. and Day, J.
1973 Teleconferencing sessions held during Dec. 1973.
- Kimbleton, S.R.
1973-74 "Network Management Information Center", A Research Program in the Field of Computer Technology, Annual Tech. Rpt.: 1973-1974 ISI/SR-74-2, USC - Information Sciences Institute, Marina Del Ray, Ca.
- Kudlick, M.D.
1973 "Network Mail Meeting Summary", RFC #469.
- Lipinski, A., Lipinski, H. and Randolph, R.
1972 "Computer Assisted Expert Interrogation: A Report on Current Methods Development", Proc. 1st ICCG, S. Winkler, ed., ACM, N.Y., p. 147.
- McKenzie, A.A., Cosell, B.P., McQuillan, J.M. and Thorpe, M.J.
1972 "The Network Control Center for the ARPA Network", Proc. 1st ICCG, S. Winkler, ed.
- Multics
1973 (a) Multics Programmers Manual, Honeywell.

(b) System Administrators Guide, Multics Programmers Manual, Honeywell.

(c) Project Administrators Guide Multics Programmers Manual, Honeywell.

Padlipsky, M.A.

1973 (a) "Two Solutions to a File Transfer Access Problem", RFC #505.

(b) "What is Free?", RFC #491.

Padlipsky, M.A., Calvin, J., Kudlick, M., Greer, C. and Crocker, D.

1974 "Design Document for a Network Performance Measurement Laboratory", USING internal memo.

Pogran, K.T.

1973 "Un-mudding Free File Transfer", RFC #501.

Sternick, H., Iseli, J.

1974 "Description of a Proposed ARPANET Help Facility", MITRE Tech. Rpt. 6723, MITRE Corp., McLean, Va.

Thomas, R.

1974 "On the Problem of Signature Authentication for Network Mail", RFC #644.

Turoff, M.

1972 "PARTY-LINE and DISCUSSION Computerized Conference Systems", Proc. 1st ICC, S. Winkler, ed., ACM.

White, J.E.

1973 "A Proposed Mail Protocol", RFC #524.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
CAC Document Number 150		
TITLE (and Subtitle) A State-of-the-Art Report on Network Data Management and Related Technology		5. TYPE OF REPORT & PERIOD COVERED Research Report-Interim
AUTHOR(s) G. Belford, S.R. Bunch, J.D. Day, et.al.		6. PERFORMING ORG. REPORT NUMBER CAC #150
PERFORMING ORGANIZATION NAME AND ADDRESS Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801		8. CONTRACT OR GRANT NUMBER(s) DCA100-75-C-0021
CONTROLLING OFFICE NAME AND ADDRESS Joint Technical Support Activity 1440 Isaac Newton Square, North Reston, Virginia 22090		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE April 1, 1975
		13. NUMBER OF PAGES 170
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
DISTRIBUTION STATEMENT (of this Report) Copies may be obtained from the National Technical Information Service Springfield, Virginia 22151		
DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) No restriction on distribution		
SUPPLEMENTARY NOTES None		
KEY WORDS (Continue on reverse side if necessary and identify by block number) Access control Computer communication networks Computer network analysis Computer network control Computer network front end Computer network measurement Computer network modeling Computer network topology Computer networks Computer protection (over)		
ABSTRACT (Continue on reverse side if necessary and identify by block number) The state-of-the-art is described in 16 areas related to data management and resource sharing on computer networks. The data management topics covered include data organization (structures and access techniques), optimization (hashing, clustering, partitioning, and compression), data languages (for structure definition and query), and file allocation on a network. The topics covered relating to the network and systems environment include communications, resource allocation, measurement and evaluation, network front ends, and security. The topics relating to network applications discuss general user support, and network management.		

block 19:

computer security
computer system measurement
data access
data base recovery
data clustering
data compression
data definition language
data integrity
data languages
data management
data organization
data partitioning
data structures
distributed computer systems
distributed data management
hashing techniques
inter-computer networks
network data management
network file allocation
network front end
network management
network protocols
network user support
packet radio
packet switching
query language
resource allocation
resource sharing

1. Report No. UIUC-CAC-DN-75-150	2.	3. Recipient's Accession No.
4. Title and Subtitle A State-of-the-Art Report on Network Data Management and Related Technology		5. Report Date April 1 1975
6.		7.
8. Author(s) G.G. Belford, S.R. Bunch, J.D. Day, et.al.		9. Performing Organization Rept. No. CAC #150
10. Performing Organization Name and Address Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801		11. Project/Task/Work Unit No.
12. Sponsoring Organization Name and Address Joint Technical Support Activity 11440 Isaac Newton Square, North Reston, Virginia 22090		13. Contract/Grant No. DCA100-75-C-0021
14. Type of Report & Period Covered Research-Interim		15.
16. Supplementary Notes		

Abstracts

The state-of-the-art is described in 16 areas related to data management and resource sharing on computer networks. The data management topics covered include data organization (structures and access techniques), optimization (hashing, clustering, partitioning, and compression), data languages (for structure definition and query), and file allocation on a network. The topics covered relating to the network and systems environment include communications, resource allocation, measurement and evaluation, network front ends, and security. The topics relating to network applications discuss general user support, and network management.

Keywords and Document Analysis. 17a. Descriptors

access control	computer security
computer communication networks	computer system measurement
computer network analysis	data access
computer network control	data base recovery
computer network front end	data clustering
computer network measurement	data compression
computer network modeling	data definition language
computer network topology	data integrity
computer networks	data languages
computer protection	data management

Identifiers/Open-Ended Terms

(continued on reverse)

SATI Field/Group

Availability Statement No Restriction on Distribution Available from the National Technical Information Service, Springfield, Virginia 22151	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 170
	20. Security Class (This Page) UNCLASSIFIED	22. Price

block 17:

- data organization
- data partitioning
- data structures
- distributed computer systems
- distributed data management
- hashing techniques
- inter-computer networks
- network data management
- network file allocation
- network front end
- network management
- network protocols
- network user support
- packet radio
- packet switching
- query language
- resource allocation
- resource sharing

Errata to State-of-the-Art Report

<u>Line</u>	<u>Correction</u>
21	For "Hsaio", read "Hsiao."
27	For "Hsaio", read "Hsiao."
7	For "Hsaio", read "Hsiao."
12	For "Hsaio", read "Hsiao."
16	For "Gottlieb", read "Gotlieb."
9	For " $n \leq 1$ ", read " $n > 1$."
31	For "=", read "<".
10	For "Hsaio", read "Hsiao".
18	For "CACM", read "Journal ACM"
26	To Winkler, A.J., add additional author: Dale, A.G.
24	For "[ANTS, 1973]", read "[Bouknight et al., 1973]".
45	For "Crowther et al.", read "Crowther, McQuillan, and Walden".
38	Parenthetic expression should be moved to end of previous sentence.
39	Add "(a), (b)" to Hayes and Sherman reference.
22	Add "(a)" to Hayes and Sherman reference.
40	Add "(a), (b)" to Hayes and Sherman reference.
9	For "Wilcov", read "Wilkov".
19	Add "(b)" to Frank reference.
10	For "an adaptive routing strategy routes", read "adaptive routing strategies as described above route".
39	Add "(b)" to Hayes and Sherman reference.
22	For "Frank and Chou; 1972, 1973" read "Frank and Chou, 1972, 1974".
22	Add "(b)" to Abramson reference.

<u>Page</u>	<u>Line</u>	<u>Correction</u>
79	10	Add "(a)" to Abramson reference.
	31	For "Fralick and Barrett", read "Fralick and Garrett".
80	26	Add "(a)" to Abramson reference.
81	27	For "Crowther, 1973", read "Crowther et al., 1973".
	30	In "Roberts, 1973 (c)", delete "(c)".
83	39	Add title: "A Protocol for Packet Network Inter-communication".
	7	For "p. 695", read "pp. 695-702."
98	43	For "mose", read "most".
145	15	Insert comma between "terminal-to-terminal" and "on-line".



UNIVERSITY OF ILLINOIS-URBANA

510.841L63C C001
CAC DOCUMENTS\$URBANA
149-151 1974-75



3 0112 007263871